Improving Breast Cancer Biomarker Predictors from Morphology via Corresponding Gaussian Processes

by

Amir Hossein Hosseini Akbarnejad

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Biomarkers for cancer are tests performed on tumoral tissue which extract information from genes (DNA, deoxyribonucleic acid), product of genes (RNA, ribonucleic acid) and proteins. The information obtained from biomarkers (abnormal amount, strutural defect, etc.) is the basis for breast cancer diagnosis and treatment. Routinely performed Biomarker tests like IHC (immunohistochemistry which reveals protein expression) and FISH (Fluorescence in situ Hybridization which reveals DNA expression) are time-consuming and expensive and are not available in many regions of the world. With the availability of digital microscopy images, several attempts have been made to apply machine learning to predict biomarker information merely from morphology (i.e. from H&E-stained histopathology images). Doing so accurately, if achievable, can solve the aforementioned issues of the biomarker tests.

In the aforementioned task current machine learning methods have low prediction performances (around 80 in terms of AUC, area under the curve) because of unavailability of large datasets. To tackle this issue we created an in-house dataset called IHC4BC containing more than 180,000 images. Thanks to the large dataset, we showed that standard machine learning methods can achieve around 90 AUCs. Moreover, we showed that weakly-supervised training with patient-level labels is not successful and the acquired patch-level labels in our proposed IHC4BC dataset have been essential to achieve high prediction performances.

Despite the good prediction performance of the obtained classifiers, our experiments showed that a high patch-level prediction performance does not mean that a method has successfully localized all relevant tissue regions, with "relevant" defined as tissue regions in which a particular gene is

over-expressed, *e.g.*, HER2 (the Human Epidermal Growth Factor Receptor 2) which is found in approximately 20% of all breast cancers and associated with a sinister outcome if not identified and properly treated. Given this limitation of methods in localizing relevant tissue regions, we manually marked near 900K HER2-positive points on the HER2 subset of the IHC4BC dataset. These manually-marked points were used to train a strongly-supervised classifier with pixel-level labels and a state-of-the-art localization method. In our analysis automatic localization is competitive to pixel-level supervision, and - intriguingly - sometimes even works better. Importantly, our analysis motivates the adoption of automatic localization with, e.g., 3K by 3K level labels specially for heterogeneous biomarkers for whom acquiring pixel-level label is not possible.

Although the proposed IHC4BC dataset enabled the classifiers to achieve high prediction performances, there are failure cases and there are a lot of research questions to be answered. To this end, we proposed a method called GPEX (**G**aussian **P**rocesses for **EX**plainning ANNs) to interpret artificial neural networks: it provides reliable explanations by performing knowledge distillation between artificial neural networks and GPs (Gaussian processes). Using our proposed GPEX we obtain Gaussian processes which are equivalent to trained neural networks. We showed that the obtained GPs can provide insight about the underlying mechanism of neural network classifiers trained on publicly available image datasets. An important goal is to identify tissue types which are missing in our in-house IHC4BC dataset and to add those images to the dataset. This goal is fulfilled in the setting known as active learning where a pool of unlabeled instances are available, and an active learner picks up some instances in the pool and asks for their labels. The active learner is supposed to pick up pool instances which are the most beneficial to a predictor. We applied GPEX to the HER2 subsubet of our IHC4BC dataset, and showed that in Bayesian active learning the GPs obtained by our proposed GPEX are a better choice than the commonly-used dropout and can improve a state-of-the-art Bayesian active learner.

Preface

This thesis was submitted as partial fulfillment of the degree Doctor of Philosophy (Ph.D.) in Computing Science at the University of Alberta. The thesis is an original work by Amir Hossein Hosseini Akbarnejad and the presented work was accomplished between September 2019 and September 2024. The research project, of which this thesis is a part, received research ethics approval (HREBA.CC-19-0347) from the Health Ethics Board of Alberta.

Material for this thesis is based on the following papers:

- Chapter 4
  - **A. Akbarnejad**, N. Ray and G. Bigras, "Deep Fisher Vector Coding For Whole Slide Image Classification," 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), Nice, France, 2021, pp. 243-246, doi: 10.1109/ISBI48211.2021.9433836.
  - N. Guruprasad, **A. Akbarnejad**, G. Bigras, P. J. Barnes and N. Ray, "A Closer Look at Weak Supervision's Limitations in WSI Recurrence Score Prediction," 2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Istanbul, Turkiye, 2023, pp. 1941-1946, doi: 10.1109 BIBM58861.2023.10385410.
  - Y. Yang, **A. Akbarnejad**, N. Ray and G. Bigras, "Double adversarial domain adaptation for whole-slide-imageclassification," Medical Imaging with Deep Learning (MIDL) 2021.
- Chapter 5
  - **A. Akbarnejad**, N. Ray, P. J. Barnes and G. Bigras, "Predicting Ki67, ER, PR, and HER2 Statuses from H&E-stained Breast Cancer Images," arxiv preprint 2308.01982, Under review in Applied Immunohistochemistry & Molecular Morphology.
- Chapter 6
  - **A Akbarnejad**, G Bigras and N Ray, "GPEX, a Framework for Interpreting Artificial Neural Networks," Conference on Neural Information Processing Systems (NeurIPS) 2023.

Acknowledgements

I would like to express my sincere gratitude to my advisors Prof. Nilanjan Ray and Dr. Gilbert Bigras for providing tremendous support and guidance in my Ph.D. study and related research. I would like to thank again Dr. Bigras and his colleagues in DynaLIFE Medical Laboratory (Edmonton, Canada) for providing and curating data for this study.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Ab: Antibody
AI: Artificial Intelligence
ANN: Artificial Neural Network
API: Application Programming Interface
AUC: Area Under the Curve
AUROC: Area Under the Receiver Operating Characteristics curve
BED: Bayesian Experimental Design
BRAF: A gene that encodes a protein regulating cell growth
CNN: Convolutional Neural Network
DAB: 3,3-diaminobenzidine, is a chromogenic substrate used in immunohistochemistry
DCIS: Ductal carcinoma in situ
ELBO: Evidence Lower Bound
EM: Expectation Maximization
ER: A protein: Estrogen Receptor
FDA: Food and Drug Administration
FFPE: Formalin Fixed Paraffin Embedded
FISH: Fluorescence in situ hybridization
GAN: Generative Adversarial Network
GP: Gaussian Process
GPU: Graphics processing unit
H&E: Hematoxylin and eosin
HER2: A protein: Human Epidermal Growth Factor Receptor 2
I/O: Input output
IHC: Immunohistochemistry
IO: Input output
Ki67: A protein which indicates when a cell is entering the cell cycle division
LSTM: Long short-term memory
MIL: Multiple-instance learning

MRI: Magnetic resonance imaging
MSI: Microsatellite instability, the condition of genetic hypermutability
NRAS: a gene that encodes a protein involved in signaling pathways that regulate cell growth and division
NTRK: A family of genes that encode neurotrophic tyrosine receptor kinases. These proteins are involved in signaling pathways that regulate cell growth, survival, and differentiation.
PCR: Polymerase Chain Reaction is a laboratory technique used to amplify specific DNA sequences, making millions of copies of a particular segment of DNA
PD-L1: (Programmed Death-Ligand 1) is a protein that helps regulate the immune system by inhibiting immune responses
PR: A protein: Progesterone receptor
RAM: Random-access memory
RNA: Ribonucleic acid
SVM: Support vector machine
TCGA: The cancer genome atlas
TILs: Tumor-infiltrating lymphocytes, a immune cell that have migrated into a tumor
TMA: Tissue micro array
TMB: Tumor mutation burden, the approximate amount of gene mutation that occurs in the genome of a cancer cell
WSI: Whole-slide image

Chapter 1

Introduction to Anatomic Pathology Methodology Applied to Cancer

In this chapter we provide basic information about anatomic pathology and the medical challenges which were addressed by this thesis through the development of novel concepts in the machine learning field. More than one century after the beginning of microscopic examination of tissue, histopathology is still the gold-standard tool for cancer diagnosis, prognosis, and treatment. In Sec. 1.1 we explain general steps to obtain a thin section (4 microns) of tissue from a 3D piece of tissue amenable to microscopic examination. Such thin sections are colorless and we'll also explain these sections are stained to provide a variety of morphological details under the microscope. In Sec. 1.2 we introduce the H&E staining which is the universal and standard stain which stains the nuclei as purple and the cytoplasm as pink. Pathologists use H&E staining to appreciate, at low magnification the two basic types of tissular compartments that are the epithelial compartment (skin, mucosa, milk ducts, etc.) and the stromal compartment (vessels, fat tissue, muscle, etc). At higher magnification, H&E can provide sub-cellular information such as nuclear texture, nucleoli, cytoplasm etc. Since approximately 1980, new techniques have revolutionized the field of anatomic pathology. Thanks to techniques like IHC (immunohistochemistry) and CISH (chromogenic in situ hybridization) one can add, to the morphology, molecular spatial information: IHC can reveal any protein and CISH any gene (DNA). New proteins and genes detection are constantly added to the armada of Biomarkers which especially improve the therapy guidance in cancer. Sec. 1.3 introduces the two aforememtioned staining techniques that are IHC) and CISH. In Sec. 1.4 we briefly introduce digital pathology, which enables the application of machine learning to histopathological images. Finally, in Sec. 1.5 we introduce OncotypeDX® recurrence-score for breast cancer, which prompted biomarker status prediction in this thesis.

1.1 Tissue Preparation

Given a tissue sample, the laboratory staff takes the steps illustrated in Fig. 1.1 to produce an histological slide which can be evaluated under microscope by pathologists.

- Step 1, Tissue sample: There are different modalities : Biopsy procedure (needle core biopsy); Surgery procedure (surgical specimen: stomach, lung, etc.).
- Step 2, Fixation and processing: the specimen is placed in a liquid fixing agent (fixative) during at least 8 hours [21] which is virtually always formalin. Formalin cross-links proteins, which helps to maintain their structure and prevent degradation.
- Step 3, The tissue is processed as follows: every single molecule of water within the tissue is removed and replaced by paraffin. Ultimately the tissue is embedded within a mould containing melted paraffin: the mould is cooled down and the final result is a FFPE (formalin-fixed, paraffin-embedded) block that can be sectioned.
- Step 4, The FFPE block is cut to make slides: thin sections of 4 microns are obtained and placed on a glass slide.
- Step 5, Slide staining: Slices from tissue are colorless and must be stained to reveal morphological details: the paraffin is chemically removed from the 4 microns section, the tissue is re-hydrated to allow the H&E stain to be performed.
- Step 6, Microscopic evaluation: Stained slide is evaluated under microscope by pathologists. This evaluation is the gold-standard for cancer diagnosis and treatment.

1.2 H&E Staining

H&E staining (short for Hematoxylin and Eosin staining) stains the cell nuclei as purple and the cytoplasm as pink. H&E staining is the gold-standard, meaning that any collected tissue is stained with H&E for the initial diagnosis. Figs. 1.2 and 1.3 illustrate some H&E images. In these images the cell nuclei are visible as purple circles/dots and extra-cellular material are visible in pink. Fig. 1.2 illustrates H&E-stained images from bone (a), breast (b), lung (c), muscle (d), and skin (e). Fig. 1.3 demonstrates an exemplary use case of H&E stained images: Gleason grading of prostate cancer. For more details please refer to the caption of Figs. 1.2 and 1.3.

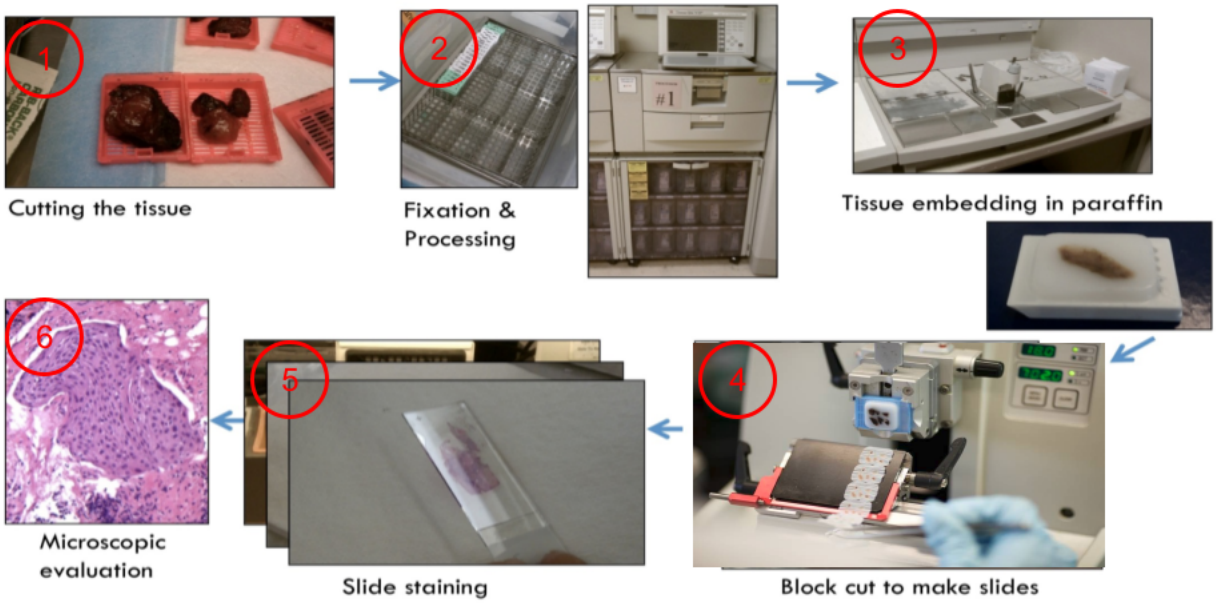
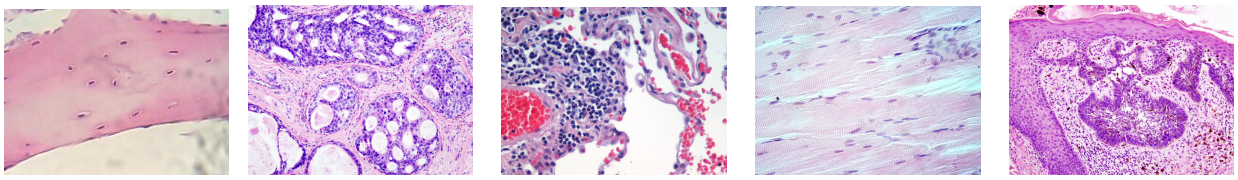


Figure 1.1: Given tissue sample (step 1), these steps are taken to obtain a slide which is evaluated by pathologists under microscope (step 6). Images borrowed and modified from [12] and [15].



(a) (b) (c) (d) (e)

Figure 1.2: Examples of H&E-stained images under microscope. Images are borrowed from [16]. In all images the cell nuclei are visible as purple circles/dots and extra-cellular material are visible in pink. a) Bone, b) DCIS (Ductal carcinoma in situ) in breast cancer. c) Lung tissue, d) Mucle tissue, and e) Skin, basal cell carcinoma.

1.3 Biomarkers

In this section we briefly describe the concept of Biomarker which can identify the statuses of gene (DNA) gene transcription (RNA) and protein expression. Afterwards in Secs. 1.3.1 and 1.3.2 we introduce two techniques for staining histopathology images, which mark either proteins or gene copies.

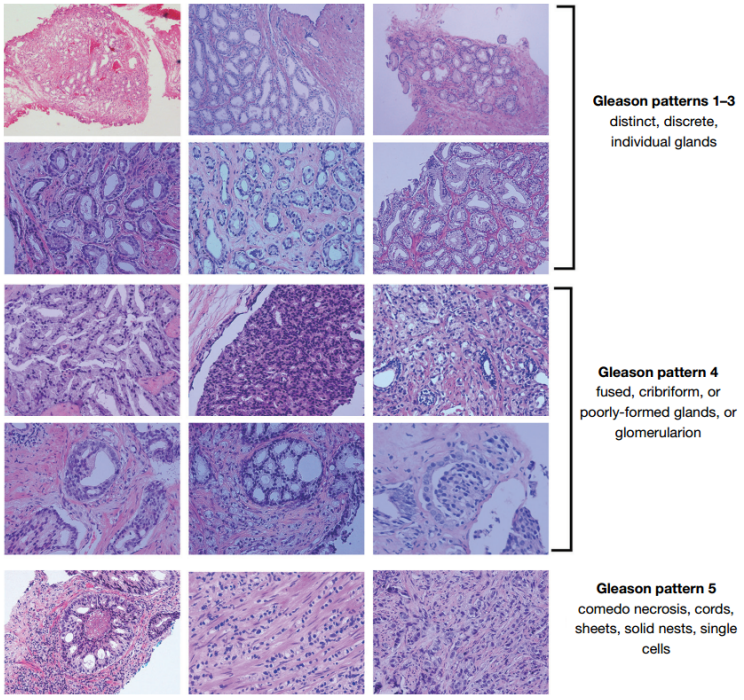
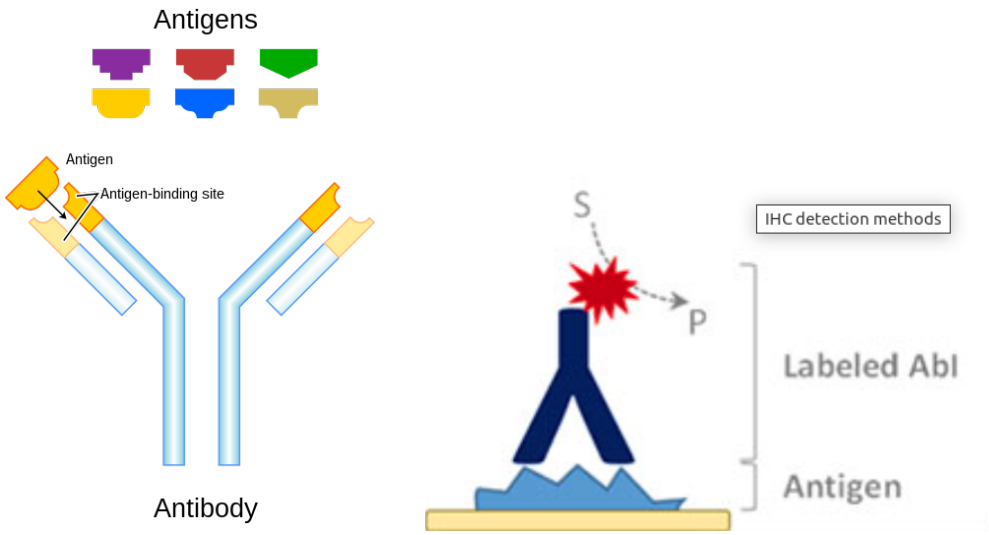


Figure 1.3: An exemplary use case of H&E-stained images is Gleason grading of prostate cancer by pathologist assessment. Regions with Gleason grade 1-3 (1st and 2nd rows) contain well-formed glands, while grade-4 regions (3rd and 4th row) contain poorly-formed glands. Finally, in grade-5 regions there are only occasional gland formations and other patterns like single cells, solid nests, etc. are visible. This image is borrowed from [43].

A Naïve Understanding from Genes, Gene Copies and Proteins

The way gene codes are executed (*i.e.* expressed) is interestingly similar to the way computer programs are executed. A computer program is initially stored on disk as a code which is not executable yet. One can think of genes as a non-executable code which is stored on disk. To execute the program, it is copied into computer's volatile memory (RAM). Gene code (DNA) is a variable sequence of four different nitrogenous bases that are Adenine, Thymine, Cytosine and Guanine. During gene expression (*i.e.* execution) a segment of DNA is firstly copied *within the nucleus*, into a RNA sequence as follows: adenine in DNA pairs with uracil in RNA, thymine in DNA pairs with adenine in RNA, cytosine pairs with guanine, and guanine pairs with cytosine.. The RNA is a messager which transfers the information outside the nucleus toward ribosomes where the code is used to produce proteins. The latter play crucial roles by serving as the building blocks of cells, enzymes that catalyze biochemical reactions and regulators of various physiological processes.



(a) (b)

Figure 1.4: a) An antibody illustrated as a big Y-shaped protein. This antibody binds only to a a specific antigen, in this figure the antigen colored in orange. b) The basic idea of the IHC technique. To mark a target protein, an antibody that binds to it is used. The antibody (shown as the Y-shaped blue protein) is equipped with a receivable signal (the red callout shape in (b)). This signal would be visible under microscope as a brown color. Figs. a and b are borrowed from [17] and [18] respectively.

1.3.1 IHC (Immunohistochemistry)

An antibody (Ab) is a large, Y-shaped protein which recognizes a unique 3-dimensional area of a given protein called the antigen. In other words, each antibody binds to a specific antigen; an interaction similar to a lock and key [17]. The binding of an antibody and an antigen is depicted in Fig. 1.4 (a). Antibodies are natural constituents of our immune system. IHC has taken advantage of this specific antigen/antibody relationship by natural antibodies obtained from mouses or rabbits exposed to a specific antigen. To mark a target protein (*i.e.* antigen), the specific antibody that binds to it is used. The antibody (shown as the Y-shaped blue protein in Fig. 1.4 (b)) is equipped with a receivable signal (the red callout shape in Fig. 1.4(b)). This signal would be visible under microscope as a brown color. There are more complicated variants of the IHC technique, and the purpose of Fig. 1.4 is to only familiarize readers with the concept.

Fig. 1.5 demonstrates how visual IHC assessment is done by pathologists. Firstly two thin tissue-slices are cut (① and ③ in Fig. 1.5). These cuts are usually a few microns apart. Afterwards, the slide from the 1st cut is stained using H&E (② in Fig.1.5) and the slide from the 2nd cut is stained using the IHC technique (④ in Fig. 1.5). The tissue-cuts obtained in ① and ③ are usually a few microns apart, and therefore they often look similar (as visible in ② and ④). Given

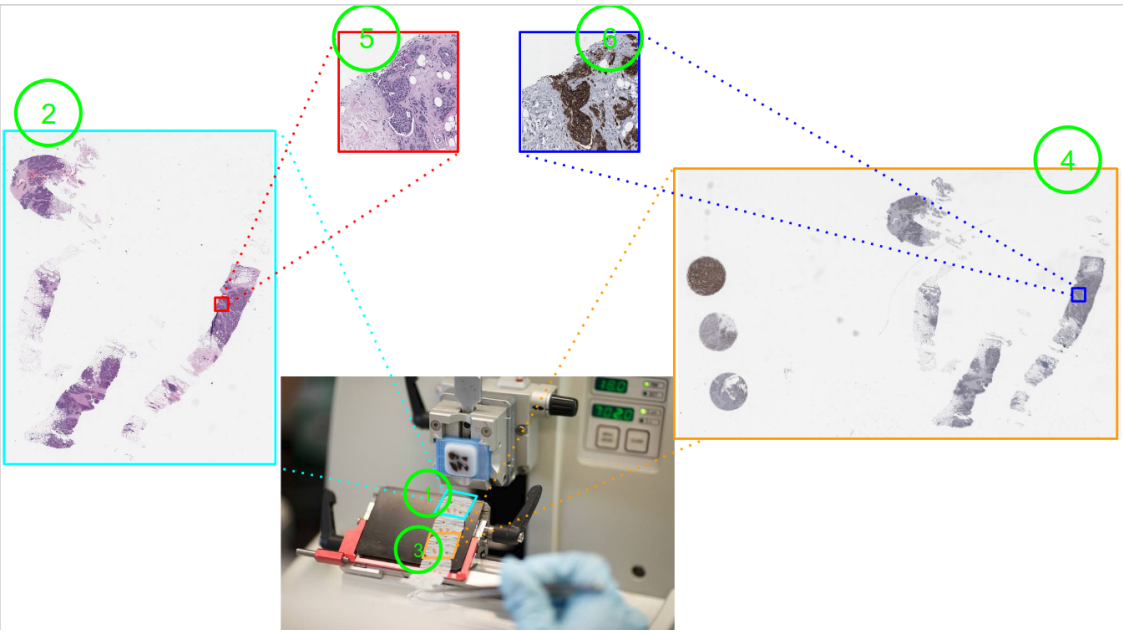


Figure 1.5: Steps taken for IHC assessment. Two tissue-slices are cut (1 and 3), one is stained with H&E (3) and one with IHC technique (4). Afterwards, for a tissue-region in the H&E modality (the red square corresponding to 5) the corresponding region in the other modality (the blue square corresponding to 6) is found. Consequently, the second modality (6) is inspected to see if brown signal is present.

a point of interest in the H&E modality (the red squared corresponding to (5)), a pathologist can find the corresponding region in the other modality (the blue square corresponding to (6)). If there is brown signal in the corresponding region (like the brown signal in (6)) the target protein exists in that tissue-region. On the other hand, the absence of brown signal indicates that the target protein doesn't exist in that tissue-region.

1.3.2 FISH (Fluorescence in situ hybridization)

The IHC technique introduced in Sec. 1.3.1 targets and marks specific proteins. Presence of proteins can indicate the expression of the corresponding gene. In this section we introduce another technique called FISH (Fluorescence in situ hybridization) which targets DNA (instead of proteins). In this technique, the probes bind to specific sequence of DNA with a high degree of complementarity (a procedure similar to the DNA transcription: instead of antigens, the utilized probes are sequences of RNA which match complementary DNA - the target) [22], By targeting DNA, FISH is a perceived as a complementary technique to IHC. The probes emit a signal which is visible under a fluorescence microscope (the red and green spots in Fig. 1.6). Chromogenic in situ hybridization (CISH) is another technique very similar to FISH. It also targets sequences of DNA using complementary

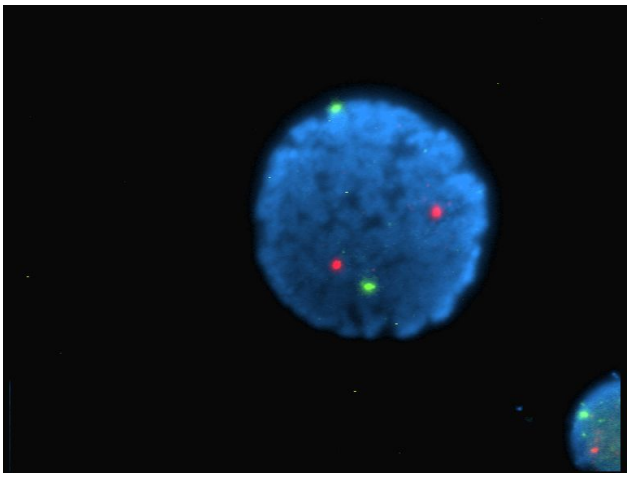


Figure 1.6: Human lymphocyte nucleus stained with the FISH technique, seen under a fluorescence microscope. The red and green spots are the signals emitted by the probes. This image is borrowed from [19].

RNA but the obtained assay can be visualized under light microscope.

1.4 Digital Pathology

1.4.1 History

Pathologists traditionally view glass slides under microscope. Digital pathology is the process of transforming histopathology slides into digital images using whole-slide scanners and subsequent analysis of these digitized images [70]. Here we borrow and summarize a short history about digital pathology from the survey of Ibrahim *et al.* [70]. Back in 1966 Prewit *et al.* [103] proposed methods to capture and analyze microscopic images. But the modern whole-slide imaging technique was introduced more recently in mid 1990s. The adoption of digitized slide images in real clinical workflow wasn't done until recently. In 2017 the seminal study by Mukhopadhyay *et al.* [96] showed that the diagnosis made by digitized WSIs (whole-slide images) is nearly as good as the diagnosis made by looking at the slides under microscope. This study was further used to get FDA approval for Philips digital pathology system [23]. Several further studies inspected the advantages and disadvantages of digital pathology. For example Mills *et al.* [93] showed that using digitized whole-slide images may initially increase pathologists' turn-around time, but this overhead in turn-around time gradually decreases to zero in a six week period.

1.4.2 Benefits

Digital pathology has several benefits. We summarize some of them here [24][70].

- 1. Turn-around time efficiency
- 2. Reduced risk of tissue loss
- 3. Enabling telepathology, *i.e.* transmission of digitized slide images and remote consulatation
- 4. Ability to register different modalities and show them side-by-side to pathologists
- 5. Enabling retrieval of histopathology images
- 6. Finally, providing machine learning methods with histopathological images

1.4.3 Virtual microscopy

The slides are scanned and stored in a format that lets pathologists view the images using computer programs. The image formats and viewer programs are designed to simulate viewing the glass slides under microscope; hence the term "virtual microscopy" [25]. Under microscope fine-grained patterns like per-nuclei information are sometimes inspected. Therefore, the slides are usually scanned in a high-resolution, resulting in huge images. A typical WSI (whole-slide image) can be 100K by 100K pixels in size and 2-3 gigabytes when stored on disk. Virtual microscopy has to simulate zooming-in and zooming-out, as done when viewing glass slides under microscope. However, resizing images on-the-fly demands a lot of computation. To avoid this computation burden, a slide image is stored in different magnification levels as shown in Fig. 1.7 so virtual microscopy softwares can directly read the image in proper magnification-level. Fig. 1.8 illustrates a whole-slide image viewed using virtual microscopy in different magnification levels.

1.5 OncotypeDX Recurrences Score for Breast Cancer

The likelihood of distant recurrence (*i.e.* - according to Oxford dictionary - the development of secondary malignant growths at a distance from a primary site of cancer) was poorly understood for specific cohorts of breast cancer patients. In 2004 Paik *et al.* [100] proposed a genomic assay for predicting the risk of distant recurrence in these patients. The output from the assay is a number between 0 and 100, which stratifies patients as follows.

- Recurrence-score between 0 and 18: low risk of distant recurrence at 10 years.
- Recurrence-score between 18 and 30: medium risk of distant recurrence at 10 years.

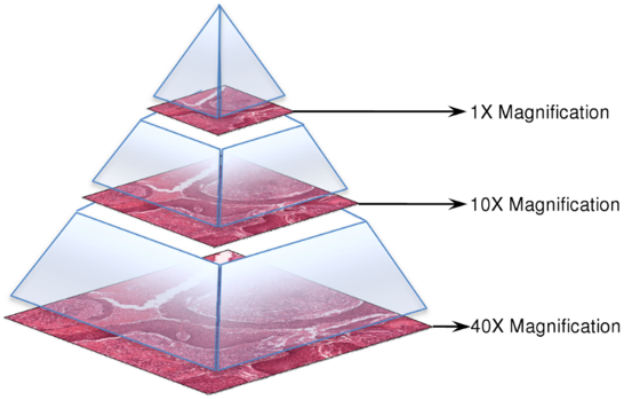
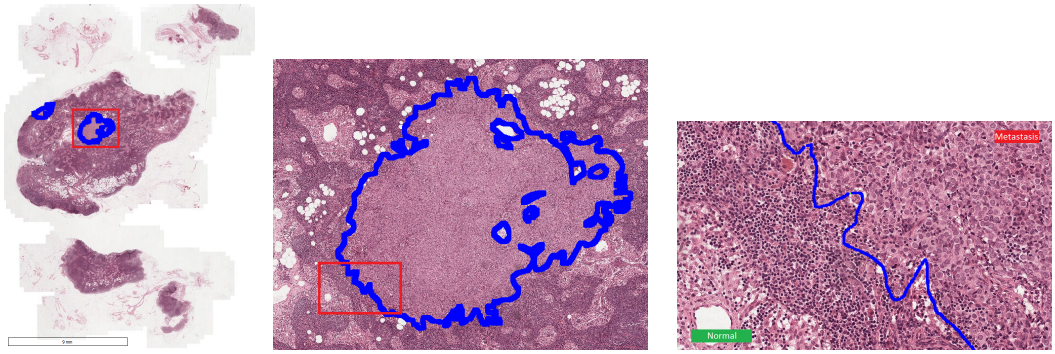


Figure 1.7: A WSI (whole-slide image) is stored as a set of images from the glass slide which are captured from different magnification levels. The image is borrowed from [20].



(a) (b) (c)

Figure 1.8: A WSI (whole-slide image) seen by a virtual microscopy software. a) The zoom-out view of the slide. b) The view is zoomed on the red rectangle in "a" and the blue polygon inside it, which appear bigger in "b". c) The view is zoomed on the red rectangle in "b". In "c" the magnification-level is high enough to see the cell nuclei. The image is borrowed from [20].

- Finally, recurrence-score higher than 30: high-risk of distant recurrence at 10 years.

The test is commercially known as OncotypeDX® recurrence-score, which is the gold-standard test to determine whether a breast-cancer patient is either a luminal-A or luminal-B breast-cancer: the former has a favorable prognosis after surgery and does not require adjuvant chemotherapy and the latter has higher risk of late recurrence and needs adjuvant chemotherapy. The study of Paik *et al.* [100] showed that the computed score is to some degree correlated with the likelihood of distant recurrence. For more information, please refer to Fig. 1.9 and its caption. The old stratification discussed above based on the thresholds 18 and 30 is replaced by a new scheme where recurrence score below 25.5 and above 25.5 are considered as low risk and high risk, respectively. For each patient the score is obtained as follows:

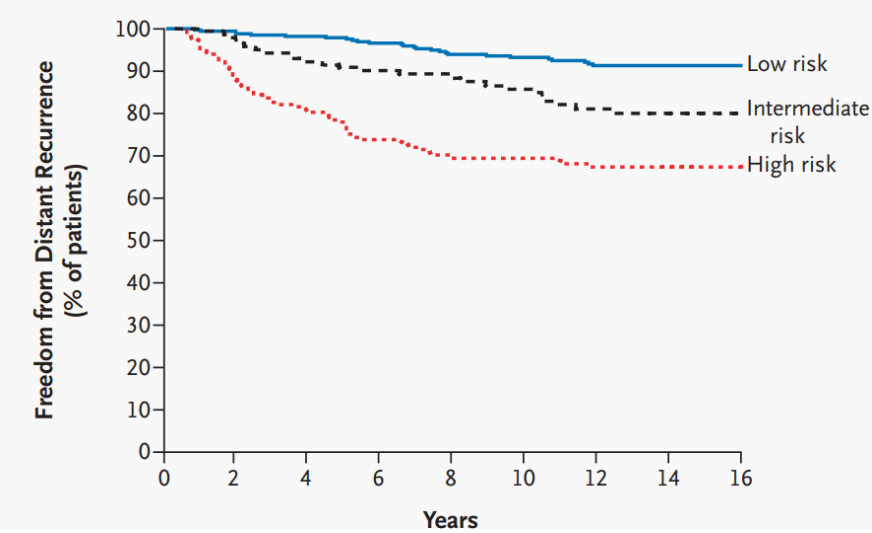


Figure 1.9: The predictive power of recurrence-score in predicting likelihood of distant recurrence in 16 years. The three curves correspond to patients with low-risk (*i.e.* recurrence-score below 18), intermediate-risk (*i.e.* recurrence-score between 18 and 30), and high-risk (*i.e.* recurrence-score above 30). This figure illustrates that more than 90% of low-risk patients are relieved from distant recurrence after 10 years. However, for high-risk patients this number is below 70%. The figure is borrowed from Paik *et al.* [100].

- 1. Best sample of tumor is selected by a pathologist and is micro-dissected to select as much as possible only the cancer component and remove the surrounding "normal" tissue within the sample.
- 2. Using RT-PCR (Polymerase Chain Reaction) RNA expression from 16 genes (16 cancer-related genes and 5 reference genes.) in 4 groups (ER, PR, proliferation, and metastatic) are quantitated and normalized with the expression of the reference gene expression.
- 3. The relative RNA quantity of the 16 cancer-related gene are inserted into a linear formula to produce a recurrence score. More precisely, the final recurrence-score is a weighted sum of the numbers obtained for the 16 genes. The weights of the linear formula are obtained in a data-driven way.

Chapter 2

Introduction

2.1 Motivation

2.1.1 Predicting Biomarker Status and Recurrence Score from H&E Images

As we discussed in Sec. 1.5 the likelihood and mechanism of distant malignant growths is not readily obvious from the morphological assessment alone: pathologists can achieve precise diagnoses.; they can also collect important data, from morphology alone, which can predict with some confidence patients' outcome: tumour histological grade, presence of tumoural emboli within small vessels, presence of tumoural deposit in regional lymph nodes, etc. But precise outcome assessment and precise therapy guidance is not properly established by a human. It is believed that tumoural morphology contains an overabundance of morphological details difficult to grasp and synthesize. This makes adoption of machine learning appealing to see if ML algorithms can accurately solve this prediction task. As for predicting biomarker status from H&E images, obtaining the H&E modality is relatively quick and cheap while using biomarkers is time consuming and expensive. Therefore, accurately predicting biomarker status only from H&E modality - if achievable - can bypass genomic tests and quickly provide supplementary gnomic information for a lot of samples and only from H&E images.

2.1.2 Interpreting Artificial Neural Networks

ANNs (Artificial neural networks) are widely adopted in machine learning. Despite their benefits, ANNs are known to be black-box to humans, meaning that their inner mechanism for making predictions is not necessarily interpretable/explainable to humans. ANN's black-box property impedes its deployment in safety-critical applications like medical imaging or autonomous driving, and makes them hard-to-troubleshoot for machine learning researchers.

2.1.3 Active Learning

In many domains - including the of field of biomedical machine learning - labels are hard and expensive to acquire. In the pool-based setting, active learning algorithms [120] (a specific case of experiment design [106]) seek the optimal samples to be labeled in a pool of unlabeled samples.

2.2 Thesis Contributions

2.2.1 Predicting Biomarker Status from H&E Images

Predicting OncotypeDX® recurrence score is not the topic of this thesis, but it motivated our follow up efforts for predicting biomarker status from H&E images.

- In an initial study, it turned out that with slide-level labels for training, the accurate prediction of recurrence score is not possible at least when roughly 1000 slides are available. Interestingly, in this setting cell-profiler features [40] outperform the state-of-the-art method CLAM [89] and our proposed method of Sec. 4 (refer to Appendix A for more information).
- The state-of-the-art WSI classification method CLAM [89] cannot achieve a high prediction performances when predicting breast cancer recurrence-score from H&E-stained WSIs. One may relate this incapability to the fact that CLAM [89] uses a resnet [60] backbone which is pretrained on imagenet [51] and is kept frozen during training. In other words, one may think if the resnet backbone is trained end-to-end, recurrence-score might be predicted way more accurately. We ruled out this hypothesis by proposing the end-to-end method of Chap. 4 and solving the practical challenges of training an end-to-end pipeline for WSI classification.
- As we discussed in Sec. 1.5, OncotypeDX® recurrence score is based on RNA quantitation from 16 cancer-related genes and 5 reference genes used as refwerence.. Now that predicting recurrence score is not possible achievable without the need to increase significantly a the WSI dataset, the natural question is: can a machine learning method predict each of those parameters used in the RS calculation separately (proliferation, ER, PR etc.)? To this end:
  - We showed that with slide-level labels and roughly 100 slides, accurately predicting each of those parameters is not possible (refer to Fig. 5.8 in Chap. 5).
  - Given the failure of training with slide-level labels, we introduced our large dataset with labels assigned to 3K by 3K images and for four major breast cancer biomarkers.
  - With our dataset we showed that it is possible to achieve prediction performances around and above 90 in terms of AUC (refer to Fig. 5.8 in Chap. 5). The results are unprecedented for this prediction task. According to the survey by Cifci *et al.* [45] (sec.

Limitations) - in the task of biomarker prediction from H&E images - the unavailability of such datasets is the main gap.

- We showed that achieving a high 3K by 3K level AUC does not mean that the predictor has correctly localized all relevant tissue regions within a 3K by 3K image (refer to Figs. 5.9, 5.10, 5.11, and 5.12).
- This motivated us to manually mark near 900K spots on HER2 positive regions and compare a strongly-supervised classifier to a automatic localization method called co-teaching [58]. Interestingly, although co-teaching has access to only patch-level (as opposed to pixel-level) labels, in most cases it is competitive to the strongly-supervised classifier. Intriguingly, in some cases co-teaching [58] performs slightly better than the strongly-supervised classifier (refer to Sec. 5.5.3 for more information).
- Although this localization ability was studied before by, *e.g.*, Laleh *et al.* [84] with slide-level labels, for 3K by 3K level labels our analysis provides novel insights. Importantly, our analysis motivates the adoption of automatic localization with, *e.g.*, 3K by 3K level labels specially for heterogeneous biomarkers for whom acquiring pixel-level label is not possible.

2.2.2 Interpreting Artificial Neural Networks

Attribution-based explanation methods like LIME[115], SHAP[90] and most gradient-based explanation methods like DeepLIFT [35] presume a linear surrogate model. Given a test instance $\boldsymbol{x_{test}}$, this simpler surrogate model is encouraged to have the same output "locally" around $\boldsymbol{x_{test}}$. Because of this "local assumptions", explanations from these methods might be unreliable, and can be easily manipulated by an adversary model [56, 112]. Moreover, these models may produce discordant explanations for a fixed model and test instance [78].

We proposed an interpretability method called GPEX with the following contributions:

- Our method finds GPs (Gaussian processes) which are globally faithful to their corresponding ANNs (artificial neural networks), hence unlike many previous methods we avoid any local assumption.
- Not many explainer models can globally match the ANN's output. Among gradient-based methods, with the best of our knowledge only Integrated Gradients [128] has a weak sense of ANN's global behaviour over the input space. Having some conditions on an ANN, another method called representer point selection [136] finds a "globally faithfull" explainer model that, similar to GPs, works with a kernel function. As we demonstrate in Sec. 6.4.4 the GP kernel that our method finds is superior due to a technical point in the formulation of representer point selection [136].

- Theoretical results on ANN-GP analogy impose some restrictions on ANNs under which the ANN will be equivalent to a GP. These conditions are too restrictive for recently used deep architectures. Moreover, those theoretical conditions need refinement as new deep architectures emerge. Nonetheless, in our proposed GPEX our formulation and method do not impose any restriction on the ANN and the method used to train it.

- Scalability is a major issue in training GPs. To address this issue, we adopted computational techniques recently used for fast spectral clustering [61] as well as a novel method to learn the GPs using mini-batches of inducing points and training instances. These computational techniques allow us to train GPs with hundreds of thousands of inducing points. According to our analysis, increasing the inducing points has been essential to get a good match between the trained GPs and ANNs. Indeed, without many inducing points GPs posterior cannot be a complex function (a function with many ups and downs [133]) and fails to match ANNs' output.

- With the best of our knowledge, our work is the first method that performs knowledge distillation between GPs and ANNs. While Borup and Anderson , [38] applied for the first time knowledge distillation to GPs, our work is distinct by distilling knowledge from an ANN to a GP. Indeed this is opposed to the self-distillation of [38] that distills knowledge from a GP to another GP.

- We implement our method as a public python library called GPEX (Gaussian Processes for EXplaining ANNs). GPEX takes in an arbitrary PyTorch module, and replaces any ANN submodule of choice by GPs. Our package makes use of GPU-accelaration, and enables effortless application of GPs without getting users involved in details of the inference procedure. GPEX can be used by machine learning researchers to interpret/troubleshoot their artificial neural networks. Moreover, GPEX can be used by researchers working on the theoretical side of ANN-GP analogy to empirically test their hypotheses.

2.2.3 Active Learning

Due to the black-box issue, the optimal active learning strategy for an ANN is not analytically known. A workaround is to apply our GPEX method to find a GP which is equivalent to the ANN thereby reducing the problem to devising an active learning strategy for a GP. To show this:

- We used GPEX in the context of Bayesian experimental design [106]. Precisely, we obtained GPs using GPEX and used them to model a hypothesis space as required by methods of Bayesian experimental design [106]. We used a Bayesian experimental design method called EPIG [126].

- We demonstrated that GPEX is a better alternative to the commonly-used dropout [127] in the context of Bayesian experimental design. We conducted experiments on synthetic data in Sec. 7.5.3, a subset of our IHC4BC HER2 dataset in Sec. 7.6.1, and a pool of whole-slide images in Sec. 7.6.2.
- Note that the benefit of using Gaussian processes for active learning has been highlighted in previous works, *e.g.* in [67] and [94]. In this thesis our goal is not to show the superiority of GPEX over those methods. Instead, our aim is to demonstrate that our GPEX - like those previous approaches - is usable in the context of active learning.

Chapter 3

Related Works and Background

3.1 Multiple Instance Learning

3.1.1 Problem Definition

In traditional supervised learning, any instance like $\boldsymbol{x} \in \mathcal{X}$ has an associated label $y \in \{1, 2, ..., C\}$. In MIL (multiple-instance learning) [37] a bag b is a set of instances: $b = \{x_1, x_2, ..., x_M\}$ and a label $y \in \{1, 2, ..., C\}$ is associated with the bag b. To be noted, in the MIL setting the bag label y is associated with the entire bag rather than any specific instance of it. Therefore, multiple-instance learning is sometimes referred to as weakly-supervised learning. In this thesis we use multiple-instance learning and weakly-supervised learning interchangeably. The whole-slide image classification task defined in Sec. 4.2 is an instance of multiple-instance learning, because one can think of a WSI (whole-slide image) as a bag containing several patches, *i.e.* "instances" in the context of multiple-instance learning [37]. The task is to predict the bag-level label (*e.g.* recurrence-score) assigned to the WSI.

The MIL setting is sometimes accompanied by an additional assumption referred to as the standard or basic MIL assumption [37]. The standard MIL assumption states that each instance is associated with a binary label $y \in \{0, 1\}$. A bag's label is 1 if and only if at least one of its instances has an associated label equal to 1. The standard MIL assumption holds for many real-world settings. For example, let the task be classifying a WSI as positive if there are tumor regions present in the WIS. In this task the standard MIL assumption holds, because a WSI's label is positive (*i.e.* contains tumor) if and only if at least one patch in it contains tumor.

3.1.2 Methods in Literature

In this section we review common themes in weakly-supervised methods for histopathology image classification. The seminal work of Ilse *et al.* [71] shows that in the MIL setting, any scoring function for bags like $S(.)$ is invariant to the permutation of bag instances if and only if the scoring function can be written as $S(b) = h\big(\sum_{x \in b} f(x)\big)$, where b is a bag, x is a member of the bag, and $f(.)$ and $h(.)$ are some functions. Given that the bag scoring function (*i.e.* the bag classifier) should be invariant to permutation in instances, the above formulation states that any such pipeline contains an instance encoder $f(.)$ followed by average pooling and a final stage $h(.)$. Despite this theoretical result, several methods with different pooling/training strategies are proposed for WSI analysis. But they usually contain two steps: 1. feature extraction, and 2. feature aggregation.

A common theme in MIL methods is spotting instances (*i.e.* patches) which are related to the bag's (*i.e.* the WSI's) label. For example C2C [125], AttMIL [71], and CLAM [89] use an explicit attention sub-module that suppresses some extracted features in the feature-aggregation stage. Hou *et al.* [66] assign a hidden binary variable to each patch, which is 1 if the patch is relevant and is 0 otherwise. Afterwards, Hou *et al.* [66] optimize the objective using expectation-minimization (EM). Zhang *et al.* [138] use a similar EM-based approach which iteratively assigns labels to patches with the additional assumption of spatial label consistency/smoothness until the label for each patch is found. Chen *et al.* [42] lift the standard MIL assumption (the assumption described in Sec. 3.1.1), and replace it with two assumptions: 1. If a WSI is in class c, at least a certain p-percent of its patches are in class c (which reproduces the pathologist's' approach when they score the grade of a tumour). 2. WSI labels form an order, similar to the case that a grade-3 WSI can contain tumor regions of grade 3, 2 or 1. A grade-2 WSI can contain tumor regions of grade 2 or 1 (but not of grade 3). A grade-1 WSI can only contain tumor regions of grade 1. Afterwards, Chen *et al.* [42] propose a training objective function that handles the aforementioned assumptions.

Another common theme in MIL methods is improving the feature extractor (*i.e.* the 1st stage of pipeline) by using self-supervised or unsupervised ideas. Since each WSI is huge, in WSI datasets there is a plethora of unlabeled patches which may benefit unsupervised methods. Sharma *et al.* [125] apply K-means clustering to patches, and during training patches are sampled in a way that each mini-batch contains patches from diverse clusters. CLAM [89] adds a term to the objective that encourages patch-level cluster formations thereby making their method data efficient. Rawat *et al.* [109] train a self-supervised feature extractor that can match left/right halves of H&E patches. They call the extracted features "fingerprints" and show that the extracted fingerprints are reasonably discriminative for predicting ER, PR, and HER2 statuses from H&E-stained whole-slide images. Tellez *et al.* [130] and Shaban *et al.* [121] adopt multi-task learning for histopathology image classification and obtain marginal improvements.

Many methods modify the aggregation stage (*i.e.* the function $h(.)$ introduced above). In

some prediction tasks if we think of a histopathology image as a bag of patches and permute the patches, this random permutation may perturb the pattern in the image. In such prediction tasks the receptive field of the pipeline should be increased so it can take in a big patch or even a WSI in its entirety. Such methods are referred to as context-aware in literature. This can be done in different ways. For example Yan *et al.* [134] use a Bidirectional Long Short-Term Memory (LSTM) in the aggregation stage. Huang *et al.* [69] split a big patch to a 3 by 4 grid. Afterwards, each element of the grid is fed to a CNN backbone to produce a feature vector. Consequently, the extracted feature vectors are concatenated and fed to a follow-up classifier. Shaban *et al.* [121] split the input image of size 1792 by 1792 to a 8 by 8 grid. Consequently, each element of the grid is fed to a CNN backbone to obtain a volumetric map. This volumetric map is further fed to some convolutional layers to produce the final output. Awan *et al.* [36] split the input image to a 3 by 4 grid. Afterwards, each element of the grid is fed to a CNN backbone to obtain a 3 by 4 volumetric map. Consequently, the volumetric map is traversed with a 2 by 2 window, and the features in any 2 by 2 window are concatenated to produce a single vector of very high dimension. This vector goes through dimensionality reduction and a follow up SVM classifier. Other ideas for implementing a context-aware aggregation stage include using vision transformers, as done by Chen *et al.* [44] or using graph convolutional networks as done by Zhou *et al.* [140]. Finally, our proposed pipeline (the pipeline explained in Chap. 4) implements the aggregation stage by Fisher vector distribution encoding.

3.2 Predicting Gnomic/Molecular Information Merely from Morphology

As underlined by Cifci *et al.* [45] (Sec. Limitations), putting aside genes with known alternations to morphology, for most genes the results in the literature are not very good. Rawat *et al.* [109] train a feature extractor for H&E images in a self-supervised way. Afterwards, they train classifiers on roughly 1000 TMA cores to predict ER, PR, and HER2 status from H&E images. They achieve slide-level AUCs of 0.89, 0.81, and 0.79 for ER, PR, and HER2 respectively. Using this sample-efficiency technique (self-supervised feature extractor) or similar ones may help improve the performance, but these techniques cannot make up for the lack of data and usually result in marginal improvement [130, 62]. Qu *et al.* [105] seek to predict point mutations and copy number alterations for some breast cancer genes and obtain 68-85 AUCs for different genes. Hohne *et al.* [63] predict BRAF mutations and NTRK gene fusions with slide-level labels, and achieve 75-85 AUCs. He *et al.* [59] use spatial transcriptomics to obtain genomic information for more than 30K H&E regions. They achieve a range of AUCs up to 90 for different genes, with most AUCs reside between 70 and 80. Moreover, for predicting the continuous expression values they achieve correlation coefficients of up to 0.50 for different genes. Zeng *et al.* [137] train a CycleGAN-like [141] virtual stainer for PR.

Zeng *et al.* [137] use unreliable labels obtained by an automatic registration method, followed by a DAB analysis that - as we will show in Fig. 5.5 - has a lot of failures. Moreover, in their work the virtual stains are only evaluated using visual criteria like SSIM (structural similarity) and PSNR (peak signal-to-noise ratio) rather than semi-quantitative methods like H-score. Jackson *et al.* [72] obtain nucleus-level labels by destaining/restaining 12 WSIs, and achieve around 90 AUCs to label each nucleus as SOX10 positive or negative. Shamai *et al.* [122] seek to predict ER, PR, and HER2 status from H&E TMA cores. The dataset contains around 5000 TMA cores and they achieved AUCs of up to 85 for PR, Ki67, and HER2 and up to 88 for ER. Tavolara *et al.* [129] seek to predict the expression of about 20000 genes in an experimental mycobacterium tuberculosis mice infection. Some of the genes (but not all) can be predicted with above 90 correlation coefficient. The gene predictors are used as an intermediate step to accurately classify each slide as supersusceptible or not-supersusceptible. In HEROHE challenge [49] the participants aimed to predict HER2 status given a training set containing 509 WSIs, and could obtains AUCs up to 84. The studies done by Anand *et al.* [33] and Liu *et al.* [87] annotate homogeneous positive or negative regions in HER2- and Ki67-stained whole-slide images (WSIs), respectively. Afterwards, the corresponding regions in H&E slides are considered positive or negative. Consequently, machine learning is applied to distinguish between positive and negative patches extracted from those homogeneous regions. The shortcoming of this approach is that in the IHC modality, many regions are associated with heterogeneous expression (especially for PR and Ki67) and all of those regions are discarded. Comiter *et al.* [48] apply two VAEs to gene expression vectors and H&E image data. Afterwards, they solve the unpaired problem via distribution matching in the VAEs' latent spaces and obtain a range of correlation coefficients for different genes. They main drawback of the study by Comiter *et al.* [48] is the limited evaluation on only four held out fixed samples throughout the study.

For the relative rare tumours with specific genomic anomaly associated with distinct morphological patterns (easily recognized by trained pathologists), the prediction performances are not surprisingly usually higher. For example for immunotherapy biomarkers like TMB, MSI, TILs or PD-L1 the AUCs reach and exceed 90 [55]. For example Shamai *et al.* [123] seek to predict PD-L1 and PD-1 status using about 5000 TMA cores. They achieve around 90 AUC for PD-L1 and a lower AUC of 85 for PD-1. As another successful example, Kim *et al.* [79] could predict mutations in BRAF and NRAS genes with AUCs higher than 90.

3.3 Interpreting Artificial Neural Networks

A common categorization is to divide interpretability methods in 1) attribution methods (including the perturbation-based methods) and 2) gradient-based methods. Notably, this categorization is not clear cut and the two aforementioned groups of methods are closely related [28].

The main idea of attribution methods is to alter the input of a neural network and observe how the output changes. Let $\boldsymbol{x_{test}}$ be an image in the test set containing M superpixels. It is computationally prohibitive to inspect the model's output for all 2^M cases of presence/absence of each superpixel. To avoid this issue, SHAP [90] assigns M values $[\phi_1, ..., \phi_M]$ to superpixels. The ϕ_j-s are called Shaply values [90] and for any subset of superpixels like $i_1, ..., i_s$ the value $(\phi_{i1} + ... + \phi_{is})$ is a good measure for the contribution of the superpixels $i_1, ..., i_s$ on the ANN's decision. The Shaply values are provably the optimal values for cooperative game theory [90]. The exact computation of SHAP values is hard, but they can be approximated. A method called LIME [115] was proposed before the adoption of SHAP values in machine learning [90]. Interestingly, it turned out that LIME [115] is one way of approximating the SHAP values. Given a test instance like $\boldsymbol{x_{test}}$, LIME [115] interprets the ANN's decision $g(\boldsymbol{x_{test}})$ by assuming that $g(.)$ is locally linear around $\boldsymbol{x_{test}}$ and takes the local linear approximation as the explanation. Although the Shaply values are provably the optimal values for cooperative game theory [90], the machine learning setting is slightly different. For example, when some superpixels are excluded from an image, it is not clear what value(s) should fill-in the excluded pixels. More importantly, since SHAP[90] considers a local explainer model based on perturbed versions of an instance, its explainations are unreliable. For instance, a model (potentially an adversary model) may behave differently on the dataset instances and the perturbed ones [112], thereby fooling the explainer model. Ghorbani *et al.* [56] argue that if the decision boundary is piece-wise linear, one can perturb a test instance to make it closer to another linear piece of the decision boundary. By doing so, a local linear explainer will pick the other linear piece of the boundary, and therefore gets fooled.

The second category of methods to be discussed here are the gradient-based explanation methods. The simple gradient method computes the gradient of output activation with respect to input pixels. In a different viewpoint, the importance of the last layer's neurons on the ANN's output is easily understood as the output is the weighted sum of the neurons in the last layer. Starting from the last layer, the simple gradient method relates the importance of the ℓ-th layer neurons to the importance of the $(\ell - 1)$-th layer neurons until it reaches the input features. More sophisticated gradient-based methods like DeepLIFT [35] address the practical limitations of the simple-gradient method, and are shown to perform better. Gradient-based explanation methods use a backpropagation-like procedure, and therefore, they are easily applicable to ANNs. An oft-said limitation is that a group of input pixels may have a negligible immediate effect (i.e. gradient) on output activations, but removing/adding those pixels simultaneously may have a large effect on output activations. Moreover, Khakzar *et al.* [78] show that most gradient-based and perturbation-based methods mistakenly give a large importance to image regions which by-design have no effect on model's output.

3.3.1 Gaussian Processes for Interpreting ANNs

The equivalence between Gaussian processes and artificial neural networks (ANNs) has received a lot of interest, because Gaussian process is a white-box model and when it matches an ANN it can unbox the blackbox of deep learning [98]. This analogy was first discovered in the seminal work of Neal [97] back in 2007, which showed that under some conditions a single-layer neural network with random weights converges to a Gaussian process. This result was further extended to ANNs with multiple layers [92] and ANNs trained with gradient descent on mean-squared error loss [74, 98]. Many works introduce sufficient conditions for making an ANN equivalent to a GP [97, 92, 98]. A common condition is that the intermediate layers have to be wide (when the intermediate layers become infinitely wide, in the limit the ANN converges to a GP). Another oft-present condition is that the activation functions should belong to a specific set of functions [92, 114]. Another exemplary condition is that Matthews *et al.* [92] require the training set to be countable (with the official mathematical definition of being "countable"), which is violated if data augmentations like random rotation or color-jitter are applied. Stack of Gaussian processes is also shown to be closely related to ANNs [32, 54, 114]. The issue is that although a Gaussian process is white-box, stack of Gaussian processes is not white-box, similar to the case of neural networks where a single linear layer is white-box, but when linear layers are stacked the ANN becomes a black box.

In the literature of GP-ANN equivalence, the neural tangent kernel [73] has been discovered more recently in 2018 by Jacot *et al.* [73] and has become popular to generate an equivalent GP. Therefore here we briefly introduce the main ideas behind neural tangent kernel and its differences to the GP kernel that we find via GPEX. For any neural network $g(.)$ with parameters $\boldsymbol{\theta}$ the neural tangent kernel is defined as follows:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \nabla_\theta g(\boldsymbol{x}; \theta)^T \ \ \nabla_\theta g(\boldsymbol{x}'; \theta), \tag{3.1}$$

where ∇ is the gradient. For a general neural network, during training the neural tangent kernel changes as the ANN changes. But interestingly, for a multi-layer perceptron whose all intermediate layers are infinitely wide and is trained on mean squared error loss, the neural tangent kernel provably is independant of the random initialization of $\boldsymbol{\theta}$ and doesn't change during training [73]. Moreover, for such an ANN we have that [98]

$$g_t(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_{1:N}) \mathcal{K}(\boldsymbol{x}_{1:N}, \boldsymbol{x}_{1:N})^{-1} \big(\boldsymbol{I}_{N\times N} - e^{-\boldsymbol{\eta}\mathcal{K}(\boldsymbol{x}_{1:N}, \boldsymbol{x}_{1:N})t}\big) \boldsymbol{y}_{1:N}, \tag{3.2}$$

where $\boldsymbol{\eta}$ is the learning rate, $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ is the training set, and t is the training iteration. Note that in Eq. 3.2 the kernel $\mathcal{K}(.,.)$ is not dependant on time. Moreover, notice that when $t \to \infty$, Eq. 3.2 becomes very similar to the equation for Gaussian process posterior mean in Eq. 6.2. Eq. 3.2 is the basis for using the neural tangent kernel for interpreting and analyzing artificial neural networks.

In particular, Eq. 3.2 can completely forecast the dynamics of a neural network during training [98]. Moreover, Eq. 3.2 makes the behaviour of the fully-trained ANN completely predictable when it sees more training data, and Mohamadi *et al.* [94] make use of this property for look-ahead active learning. The GP kernel that we find via our proposed GPEX have many distinctions to the neural tangent kernel. First of all, our GPEX method doesn't require all intermediate layers to be infinitely wide and is completely agnostic to the architecture of the ANN and the loss used to train the ANN. Moreover, the neural tangent kernel is fixed while in GPEX the kernel can vary as, *e.g.*, the ANN is trained for more iterations or more training data is added. This property makes GPEX a better choice if one aims to supervise the underlying kernel with, *e.g.*, contrastive learning.

Scalability is a major issue when training GPs, and including a few inducing points may limit the flexibility of GP's posterior [133]. Here we review some previous methods to tackle the computational challenges of training GPs. SV-DKL [111] derives a lower-bound for training a GP with a deep kernel. In this method, a grid of inducing points are considered in the kernel-space (like the vectors $\{(\tilde{\boldsymbol{u}}_m^{(\ell)}, \tilde{v}_m^{(\ell)})\}_{m=1}^M$ with the notation of this thesis). Afterwards, each input instance is firstly mapped to the kernel-space and the output is computed based on similarities to the grid points in the kernel-space. Since the GP posterior is computed via the grid points, SV-DKL [111] is scalable. But unfortunately the number of grid points cannot be increased to above 1000 even for Cifar10 [83] and with a RTX 3090 GPU. Therefore, this may limit the flexiblity of the GP's posterior [133]. A more recent framework called GPytorch [113] provides GPU acceleration. However, its computational complexity is quadratic in number of inducing points. Other approaches to improve scalability of GPs include: considering structured kernel matrices [47], kernel interpolation [132], and imposing grid-structure on including points [111]. The closest work to our scalability method in Alg. 5 is the concurrent work of Lin *et al.* [85] where they compute the output of our Alg. 5 via an iterative optimization thereby generating samples from GP posterior. However, using the approach of Lin *et al.* [85] would not work in our case since we need a differentiable procedure like Alg. 5 as opposed to an iterative solver.

Chapter 4

Deep Fisher Vector Encoding for WSI Classification

4.1 Synopsis

In this chapter we explain our proposed WSI classification method. The distinction of this method is that it is trained end-to-end and without the basic MIL assumption discussed in Sec. 3.1.1. Moreover, it is efficiently evaluated on huge WSI datasets using a subset of patches population.

4.2 Problem Definition: WSI classification

In WSI (whole-slide image) classification the dataset contains a number of WSIs. Each WSI is a huge image in the order of 100K by 100K pixels. For each WSI a ground-truth label is available. In other words, in WSI classification we have a dataset $\mathcal{D} = \{[\boldsymbol{W}_1, y_1], [\boldsymbol{W}_2, y_2], ..., [\boldsymbol{W}_{|\mathcal{D}|}, y_{|\mathcal{D}|}]\}$ where $\mathbf{W}_n$ is a WSI and y_n is a label assigned to it. The label is a categorical variable that tells, *e.g.*, whether there is a tumor region over the WSI. As another example, the label can be the result of a pathologist assessment (*e.g.* tumor grade, cancer subtype, hormonal status, etc.) on the WSI as a categorical number. To train and evaluate a machine learning method, as usual the dataset is split to the train/validation subset for training and the testing subset for evaluation.

WSI classification is notoriously a challenging problem due to many difficulties.

- 1. The size of dataset $|\mathcal{D}|$ is usually small and often less than 1000. This is a major challenge, because even if a machine learning method can encode all essential features of a WSI into a vector, the classifier placed on top of the vectors (*i.e.* WSI encodings) would have access to a small number of labels and may fail due to, *e.g.*, overfitting.

- 2. The label assigned to a WSI might be related to a small region of it, and it is the method's responsibility to localize relevant regions merely from the labels assigned to WSIs. For example in tumor detection task, the labels merely tell whether there is a tumor region "somewhere" over a WSI or not, without providing any spatial information about the location of those tumor regions. In other words, if relevant regions are sparse, localizing those regions would be like finding a needle in a haystack.

- 3. Histopathology images contain a lot of variations, which may hinder the adoption of commonly used ML methods. For example training a variational auto-encoder [81] on H&E-stained histopathology patches is challenging, and reconstructed images tend to be quite different from their corresponding inputs [5]. Or the accuracy gain from self-supervised pretraining may saturate when as low as 50K unlabeled patches are used during pretraining [46].

- 4. A WSI is typically as large as 100K by 100K pixels, and the disk size of a typical WSI dataset may well surpass one terabyte. This is a practical challenge in adoption of machine learning algorithms on WSIs both during training and during evaluation. Of note, in order to deploy an algorithm in clinical workflow the evaluation time has to be roughly 1 minute per WIS, because modern WSI scanners can scan roughly 1 WSI per minute.

4.3 Proposed Method

Let $\boldsymbol{x}$ be a random patch extracted from a WSI. Let $f(.)$ be a function that takes in $\boldsymbol{x}$ and produces a D-dimensional vector. The vector $f(\boldsymbol{x})$ is often referred to as a *descriptor*. We encode the WSI as the distribution $P\big(f(\boldsymbol{x})\big)$. To represent this distribution as a vector, we used Fisher vector distribution encoding [34]. Fisher vector distribution encoding [34] encodes a distribution $P\big(f(\boldsymbol{x})\big)$ by concatenating different statistics from the distribution. Instead of computing simple statistics like mean and variance, in this approach a set of fixed vectors $\{\boldsymbol{v_1}, ..., \boldsymbol{v_m}\}$ in the space of descriptors are considered and are used to compute some first order and second order statistics. Let $s_{ij} \in [0, 1]$ be the soft assignment of $f(\boldsymbol{x_i})$ to the fixed vector $\boldsymbol{v_j}$. A set of descriptors $\{f(\boldsymbol{x_1}), ..., f(\boldsymbol{x_n})\}$ are encoded as

$$\frac{1}{n}\sum_{i=1}^{N} FV\big(f(\boldsymbol{x_i})\ ;\ \boldsymbol{v_1}, ..., \boldsymbol{v_m}\big), \tag{4.1}$$

where $FV : \mathbb{R}^{D+m} \to \mathbb{R}^{2m}$ is a function defined as follows:

$$\begin{aligned} FV\big(f(\boldsymbol{x_i})\ ;\ \boldsymbol{v_1}, ..., \boldsymbol{v_m}\big) = \Big[&\frac{s_{i1}}{c_1}\big(f(\boldsymbol{x_i}) - \boldsymbol{v_1}\big), ..., \frac{s_{im}}{c_m}\big(f(\boldsymbol{x_i}) - \boldsymbol{v_m}\big), \\ &\frac{s_{i1}}{\hat{c}_1}\big(f(\boldsymbol{x_i}) - \boldsymbol{v_1}\big)^2, ..., \frac{s_{im}}{\hat{c}_m}\big(f(\boldsymbol{x_i}) - \boldsymbol{v_m}\big)^2\Big]. \end{aligned} \tag{4.2}$$

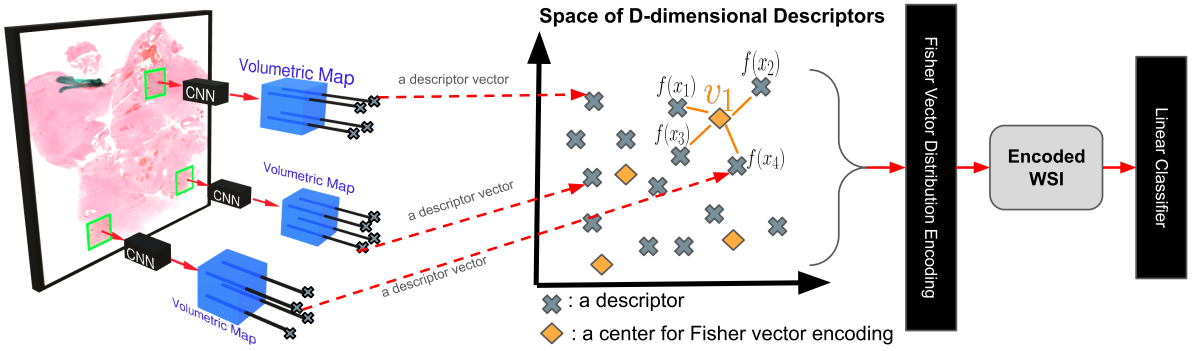


Figure 4.1: The proposed pipeline [31] based on Fisher-vector distribution encoding [34]. Some patches (the green rectangles on the left) are fed to a CNN backbone and the blue volumetric maps are obtained. Each spatial position of the volumetric maps (across the channels) defines a D-dimensional vector in the space of descriptors (the plot in the middle). Having some fixed anchors in the space of the descriptors (the orange diamonds), a vector is obtained according to Eq. 4.2. One can think of the expectation of this vector as "Encoded WSI", which is further fed to a linear classifier to produce the final predicted label for the WSI.

In Eq.4.2, c_j and $\hat{c}_j$ are some constants. We encode a WSI to a vector as follows:

$$\text{the vector encoding of a WSI} \quad = \quad \mathop{\mathbb{E}}_{\substack{x \sim otsu \\ foreground}} \Big[FV\big(f(\boldsymbol{x})\ ;\ \boldsymbol{v_1}, ..., \boldsymbol{v_m}\big) \Big], \tag{4.3}$$

where $\mathbb{E}$ denotes mathematical expectation and $\boldsymbol{x}$ is a random patch extracted from the WSI's foreground obtained by Otsu's method [99]. After encoding a WSI to a vector by Eq. 4.3, we feed this vector to a linear classifier to predict the WSI's label. During training, the expectation of Eq. 4.3 is approximated by firstly extracting a few patches $\boldsymbol{x_1}, ..., \boldsymbol{x_n}$ from the WSI (using our inhouse package PyDmed[26]) and then averaging over them as in Eq. 4.1. Therefore our pipeline is trained end-to-end and the expectation of Eq. 4.3 is computed using a minibatch of samples both during training and during testing. Fig. 4.1 illustrates the pipeline.

We implemented the function $f(\boldsymbol{x})$ by a convolutional neural network. As mentioned above, given a patch $\boldsymbol{x}$ the function $f(\boldsymbol{x})$ may produce only one descriptor in $\mathbb{R}^D$. However, inspired by the method CBoF [101], we implemented the function $f(\boldsymbol{x})$ by a fully convolutional module that produces a volumetric map of shape $[D \times V \times V]$. This volumetric map indeed contains $V \times V$ descriptors in $\mathbb{R}^D$.

	her2 contest	accuracy	precision	recall	F-score	testing-time(s)
Proposed Method	426.42(38.31)	63.33(7.64)	0.75(0.02)	0.63(0.08)	0.61(0.08)	1439.2(21.6)
Proposed Method (sweep)	430(20.64)	62.50(0.08)	0.59(0.07)	0.62(0.08)	0.58(0.07)	12158(120.05)
Baseline Average Pooling	413.21(0.17)	57.14(0.03)	0.55(0.02)	0.57(0.03)	0.52(0.03)	1410(17.2)

Table 4.1: Performance of predicting breast cancer HER2 score. Rows (resp. columns) correspond to different methods (resp. performance measures).

	auc	accuracy	precision	recall	F-score	testing-time(s)
Proposed Method	0.92(1.01)	90.82(1.02)	0.88(0.02)	0.90(0.00)	0.89(0.01)	3065(10.53)
Proposed Method (sweep)	0.91(2.02)	91.2(0.89)	0.88(0.02)	0.89(0.72)	0.88(0.01)	26406(16.16)
Baseline Average Pooling	0.90(0.54)	90.38(0.32)	0.86(0.02)	0.89(0.00)	0.87(0.01)	3073(20.50)

Table 4.2: Performance of predicting brain astrocytic (glial) tumor grade. Rows (resp. columns) correspond to different tasks (resp. performance measures).

4.4 Experiments

4.4.1 Predicting HER2 and Tumor Grade for Breast/Brain Cancer

We evaluated the method of Sec. 4.3 on 4 tasks, 2 of which are discussed in this section:

- **Predicting Breast Cancer HER2 score from IHC (*i.e.* H-DAB stained) HER2 WSIs:** we downloaded the dataset used in Warwick university HER2 contest [104]. This dataset contains 52 WSIs for training and 34 WSIs for testing. As we didn't have access to the labels for WSIs in the testing set (those labels weren't released by contest organizers), we considered 60% of the training WSIs for training and the remaining 40% for testing. We repeated the experiment for three random train/test splits of the dataset. The results are provided in Tab. 4.1.
- **Classifying brain astrocytic (glial) tumor grade from H&E whole slide images**: We downloaded H&E slides from 120 patients from TCGA [3] dataset. For each patient, we considered the H&E slide which was most recently scanned. The task was to classify astrocytomas in two groups: grade IV astrocytoma and other lower grades astrocytomas. We considered 50% of the cases for training and the remaining 50% for testing. Following the authors of [95], we used the labels reported in the supplementary material of [41]. We repeated the experiment for three random train/test splits of the dataset. The results are provided in Tab. 4.2.

Experimental Setup

For the fully convolutional module of Fig. 4.1, we used a pre-trained Resnet-50 backbone [60] followed by a convolutional layer that reduces the number of channels (i.e. the dimensionality of the descriptor space) to 10. Finally we added a batch-normalization layer to this module. In all of our experiments we set the number of Fisher vector coding centers (i.e. the variable m in Eq. 4.3) to 10. Moreover, with the notation of [34], we set the parameters of Fisher vector encoding as $\pi_k = 0.1$ and $\sigma_k = 0.1$. During training we set the batch size to 32, and we trained on each training set for 80000 iterations. We used a RMSprop optimizer with learning rate 0.00001.

Approximating the expectation of Eq. 4.3 by a few samples may result in a large noise in the approximate gradient, which in turn may cause the optimization to fail. One solution is to increase the batch size. However, doing so is impractical due to the limited memory of GPU. To tackle this issue, we used PyTorch's mechanism for accumulating gradients in a burst of backward passes and updating parameters at the end of each burst [11]. By doing so, we effectively increased the batch size to 32×20. In the test phase, we approximate the expectation of Eq. 4.3 by extracting 500 patches from each WSI. We implemented both the training and the testing phase by our developed package PyDmed [26] (Python Dataloader for MEDical imaging). For more information about the experimental setup please refer to the publicly available repository [10].

Results

As a baseline, we replaced the Fisher vector encoding stage by an average pooling layer to see how the prediction performance changes. In Tabs. 4.1 and 4.2 this baseline is referred to as *Baseline Average Pooling*. We split each dataset to training/testing sets three times. Tabs. 4.1 and 4.2 provide the average evaluation metric over these three runs. In Tabs. 4.1 and 4.2 the numbers within parenthesis denote standard deviations. In the second column we report a performance measure used in Warwick HER2 contest [104]. According to the leader board of the contest [104], our method outperforms the competitors by a large margin: we obtained the highest combined score of 430 which is higher than the highest score of 402.5 among the participants in the contest [104]. For brain tumor grading, our prediction performance is comparable to the results reported in [95], although our dataset is much smaller (120 WSIs vs. 700 WSIs).

We implemented both training and testing phases by our developed package PyDmed [26] (Python Dataloader for MEDical imaging). PyDmed tailors the idea of PyTorch's dataloader to medical datasets. With PyDmed [26] our training time on more than 2.5 million patches in about 8 hours. Moreover, our testing time is less than one minute per WSI. The reason behind this efficiency is to think of each WSI as a population of patches and to estimate the expectation of Eq. 4.3 by some sampels from the population. The alternative way is to sweep over each WSI in the testing

phase. In Tabs. 4.1 and 4.2 the latter approach is denoted by *Proposed Method (sweep)*. According to Tabs. 4.1 and 4.2, this approach dramatically increases the testing time and the prediction performance remains almost the same.

4.4.2 Unsupervised Domain Adaptation and Recurrence Score Prediction

Besides the two tasks described in Sec. 4.4.1, the proposed method was used in our other studies.

- Double-adversarial unsupervised domain adaption was applied on this pipeline, and the results were published [135].
- We also adopted this pipeline for predicting breast cancer recurrence-score from H&E-stained whole-slide images. The experiments were mainly done by Ms. Namitha Guruprasad (the first author of [57]) and the author of this thesis took part in curating the splits as well as implementing the cell-profiler baseline. The results are provided in Appendix A. Interestingly according to Fig. A.1 the proposed method of Sec. 4.3 outperforms the state-of-the-art method CLAM [89]. Moreover, according to Fig. A.1 the cell-profiler [40] baseline outperforms both CLAM [89] and our proposed method that we described in this chapter.

4.5 Conclusion

In sum, in this section we presented our proposed method for WSI classification, with the following major findings:

- The state-of-the-art WSI classification method CLAM [89] cannot achieve a high prediction performances when predicting breast cancer recurrence-score from H&E-stained WSIs. One may relate this incapability to the fact that CLAM [89] uses a resnet [60] backbone which is pretrained on imagenet [51] and is kept frozen during training. In other words, one may think if the resnet backbone is trained end-to-end, recurrence-score might be predicted way more accurately. But we ruled out this hypothesis by proposing the end-to-end method of Sec. 4.3 and solving the practical challenges of training an end-to-end pipeline for WSI classification.
- We showed that our proposed pipeline surpasses the participants of Warwick university HER2 contest [104] in HER2 prediction, performs on-par with the method by Momeni *et al.* [95] in brain tumor grading, and outperforms CLAM [89] in predicting breast cancer recurrence-score.

Chapter 5

The Proposed Dataset for Predicting Breast Cancer Biomarker Status

5.1 Synopsis

In this section we present our dataset for predicting the status of four biomarkers (Ki67, ER, PR, and HER2) from H&E-stained breast cancer tissue images, and use it to achieve good prediction performances. As confirmed by other studies, the lack of such a dataset was the main gap for this prediction task.

5.2 Overview and Motivation

Training a WSI classifier with WSI-level labels could achieve impressive performances in predicting labels which are related to visual patterns [89, 86]; visually identifiable patterns like metastatic or tumor tissue regions. However, in the case of our recurrence-score prediction task with a dataset of roughly 700 WSIs, WSI classification with WSI-level labels is not effective - as seen in the experiments of Sec. 4.4.2 (the results of Fig. A.1) and also the experiments in Guruprasad *et al.* [57]. This incapability of training with WSI-level molecular (*i.e.* non-visual) labels has also been reported in other studies, *e.g.*, in the abstract by Shamai *et al.* [124] for predicting recurrence-score or in the comprehensive benchmarking by Laleh *et al.* [84]. These facts motivated us to break down the recurrence-score prediction problem as follows.

- The OncotypeDX® recurrence-score is a weighted sum of many biomarker statuses, four of which have a relatively large importance weight: Ki67, ER, PR, and HER2. So instead of directly predicting the recurrence-score for each WSI, one may think predicting each biomarker

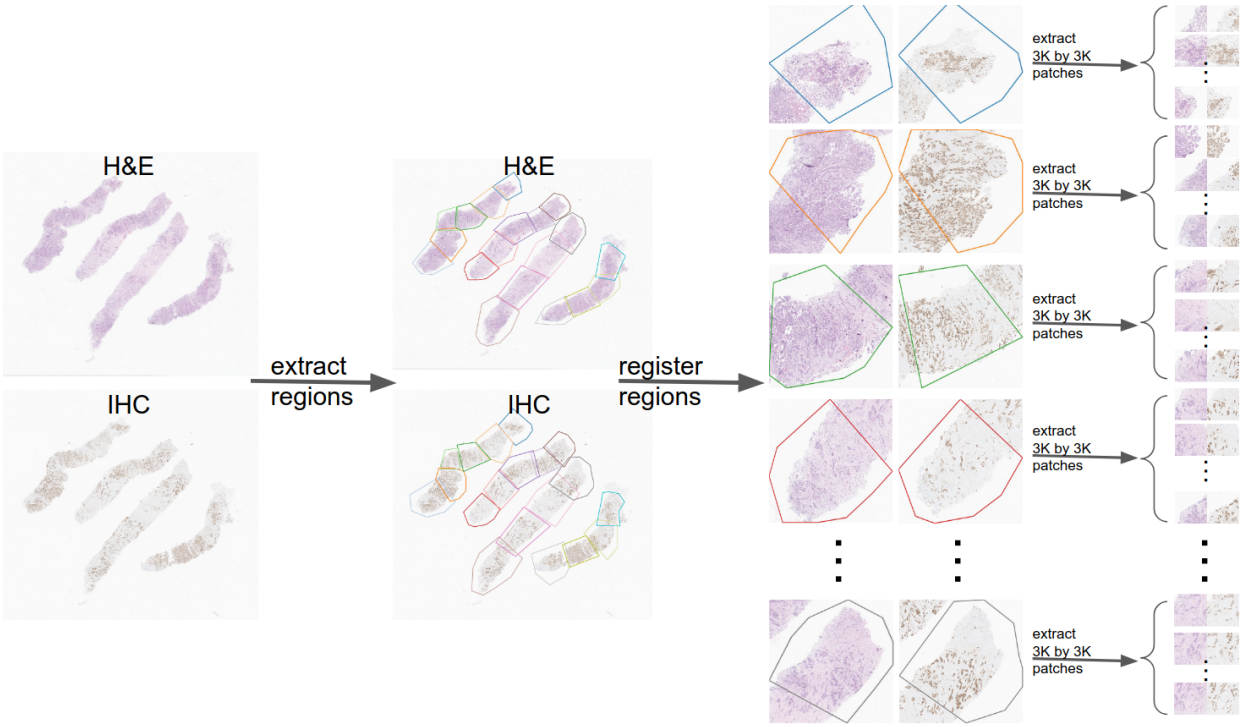


Figure 5.1: The steps taken to obtain 3K by 3K H&E-IHC pairs from a WSI-pair. Details are provided in Sec. 5.3.2.

status separately is easier.

- As we discussed above, training a WSI classifier with WSI-level labels may not be effective for biomarker-related labels. Therefore we collected a dataset containing labels which are related to a 3K by 3K patch, rather than a huge whole-slide image.

All in all, our experiments showed that training with WSI-level labels is not effective in predicting each biomarker, but using a large dataset with fine-grained labels results in impressive prediction performances.

5.3 The Dataset

5.3.1 Tissue and Slide Preparation

All ER, PR, HER2 and Ki67 IHC preparations were performed on unselected sequential 50 breast biopsies collected in 2022 with strict pre-analytical and analytical controls consisting of 1) absence of cold ischemia (virtually all biopsy tissue put immediately in formalin fixative after collection), 2) minimum of 24 hours and a maximum of 48 hours in formalin, 3) utilization of on-slide controls

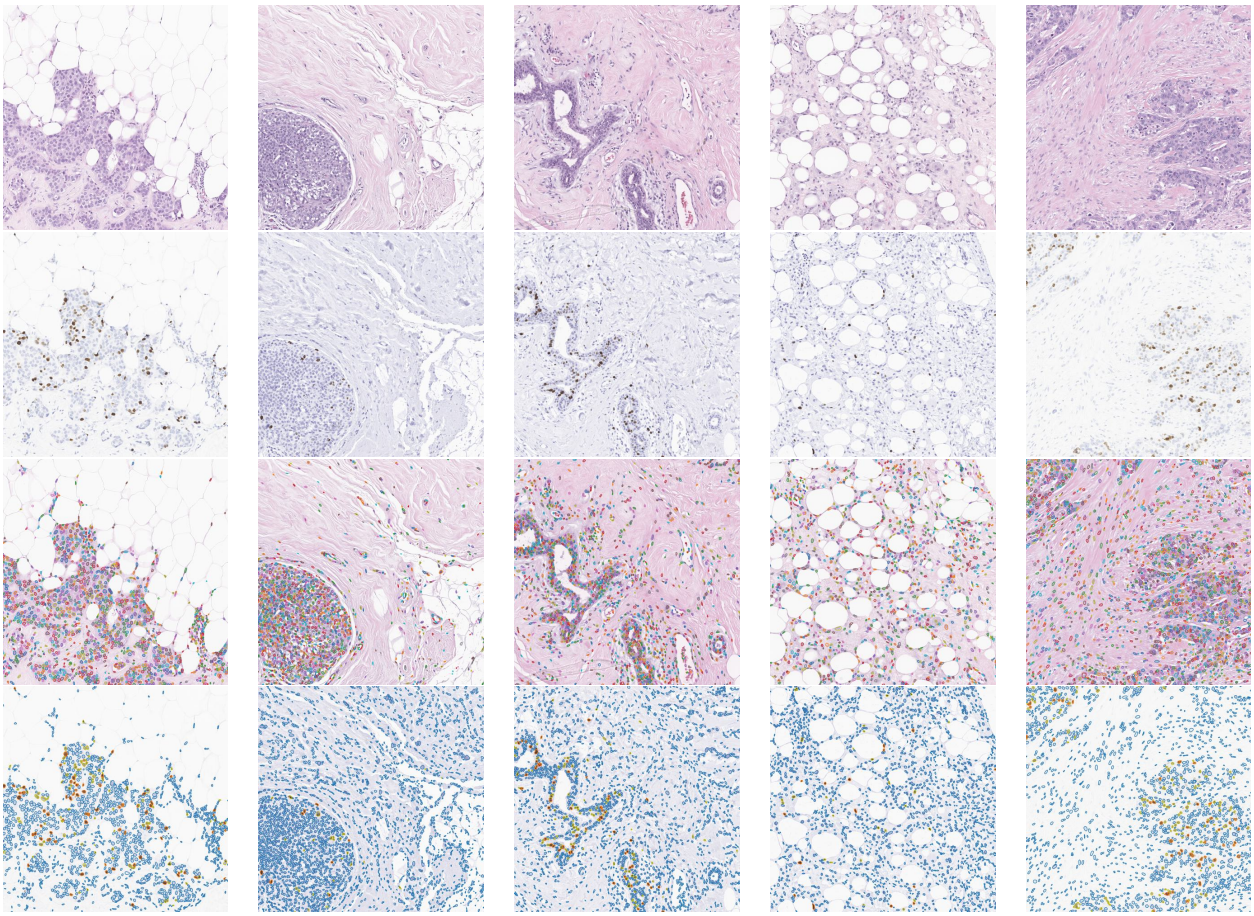


Figure 5.2: Examples of H&E-Ki67 pairs in the dataset. Each column corresponds to a H&E-IHC pair. Row1: H&E, Row2: IHC, Row3:nuclei sementations on H&E, Row4: the result of DAB-analysis on IHC where blue, yellow, orange, and red colors mark 0 (*i.e.* negative), 1+ (*i.e.* weakly stained), 2+ (*i.e.* moderately stained), and 3+ (*i.e.* strongly stained) nuclei, respectively.

to ensure successful IHC performance by the IHC instrument and 4) repeating confirmatory IHC procedure when finding negative tumoral ER and PR in the absence of positive internal controls (normal mammary ducts). The IHC clones utilized for ER, PR, HER2 and Ki67 were respectively SP1, PgR 636, SP3 and MIB-1. These IHCs assays were obtained using the automated platforms: Ventana Ultra (for ER, PR and HER2) and Dako Omnis (Ki67). The slides were scanned at 40X using the Aperio GT 450 - Automated, High Capacity Digital Pathology Slide Scanner.

5.3.2 Obtaining Pairs of Patches

Fig. 5.1 illustrates how we extract pairs of patches given a pair of H&E and IHC whole-slide images. Given variability of tissue disposition on glass slide, a single rigid transformation cannot perfectly register a WSI pair. Therefore, we firstly annotated region-pairs from two given matched

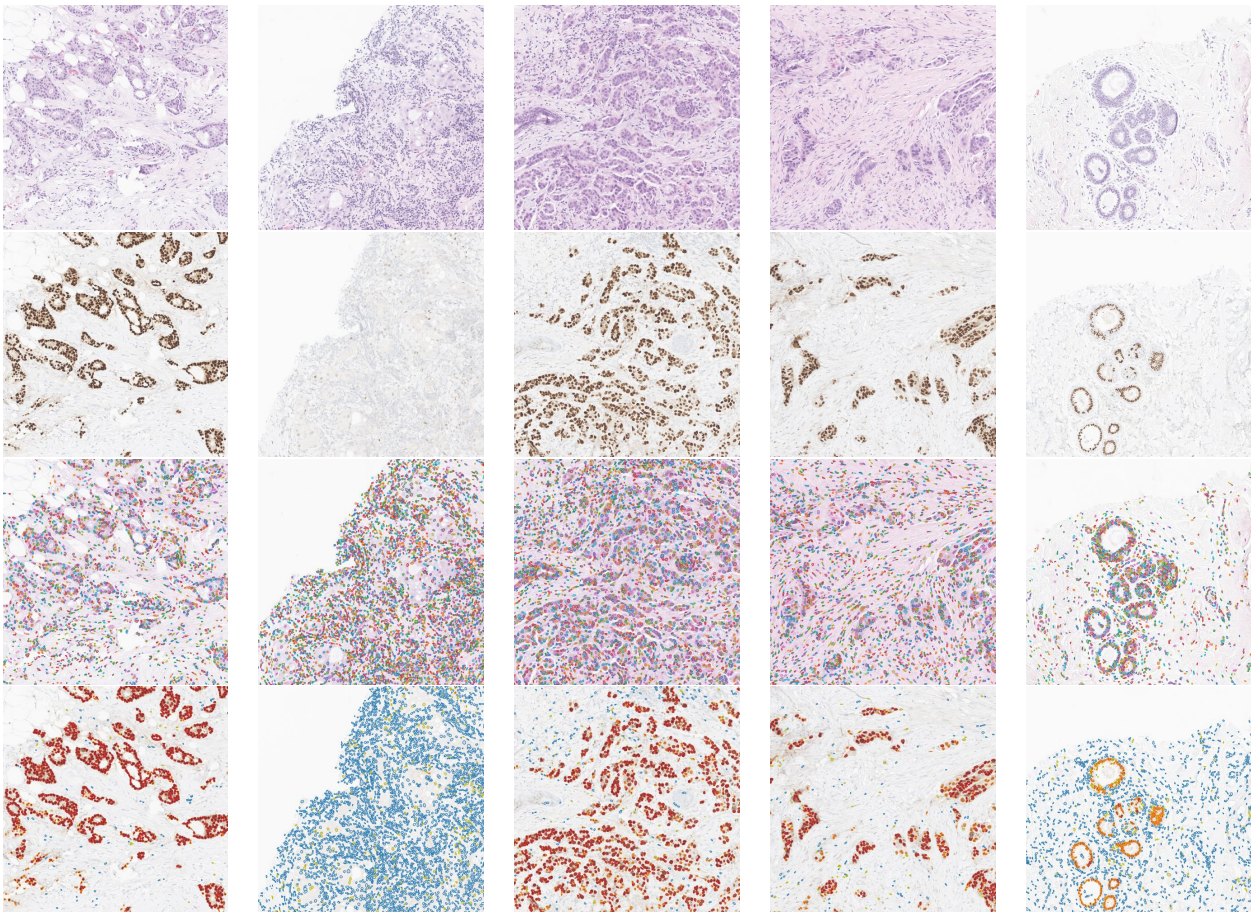


Figure 5.3: Examples of H&E-ER pairs in the dataset. Each column corresponds to a H&E-IHC pair. Row 1: H&E, Row 2: IHC, Row 3:nuclei sementations on H&E, Row 4: the result of DAB-analysis on IHC where blue, yellow, orange, and red colors mark 0 (*i.e.* negative), 1+ (*i.e.* weakly stained), 2+ (*i.e.* moderately stained), and 3+ (*i.e.* strongly stained) nuclei, respectively. In the first and fourth columns there are several nuclei which are detected in the H&E modality (row 3) but they are missed in the IHC modality (row 4). Moreover, in the fifth column in the IHC modality (row 4) the number of nuclei are overestimated.

WSIs. In Fig. 5.1 this step is labeled as "extract regions". Afterwards, we manually registered each region-pair. In Fig. 5.1 this step is labeled as "register regions". Finally, we traversed each region-pair with a stride of 1500 and extracted patch-pairs. The final result of this step is a set of patch-pairs each of which are 3000 by 3000 (the pairs shown in the right side of Fig. 5.1).

5.3.3 H-DAB Analysis to Obtain Labels

Each pair contains a H&E patch of size 3000 by 3000 and the corresponding IHC patch of size 3000 by 3000. We ran H-DAB analysis on each IHC image to obtain marker information for the

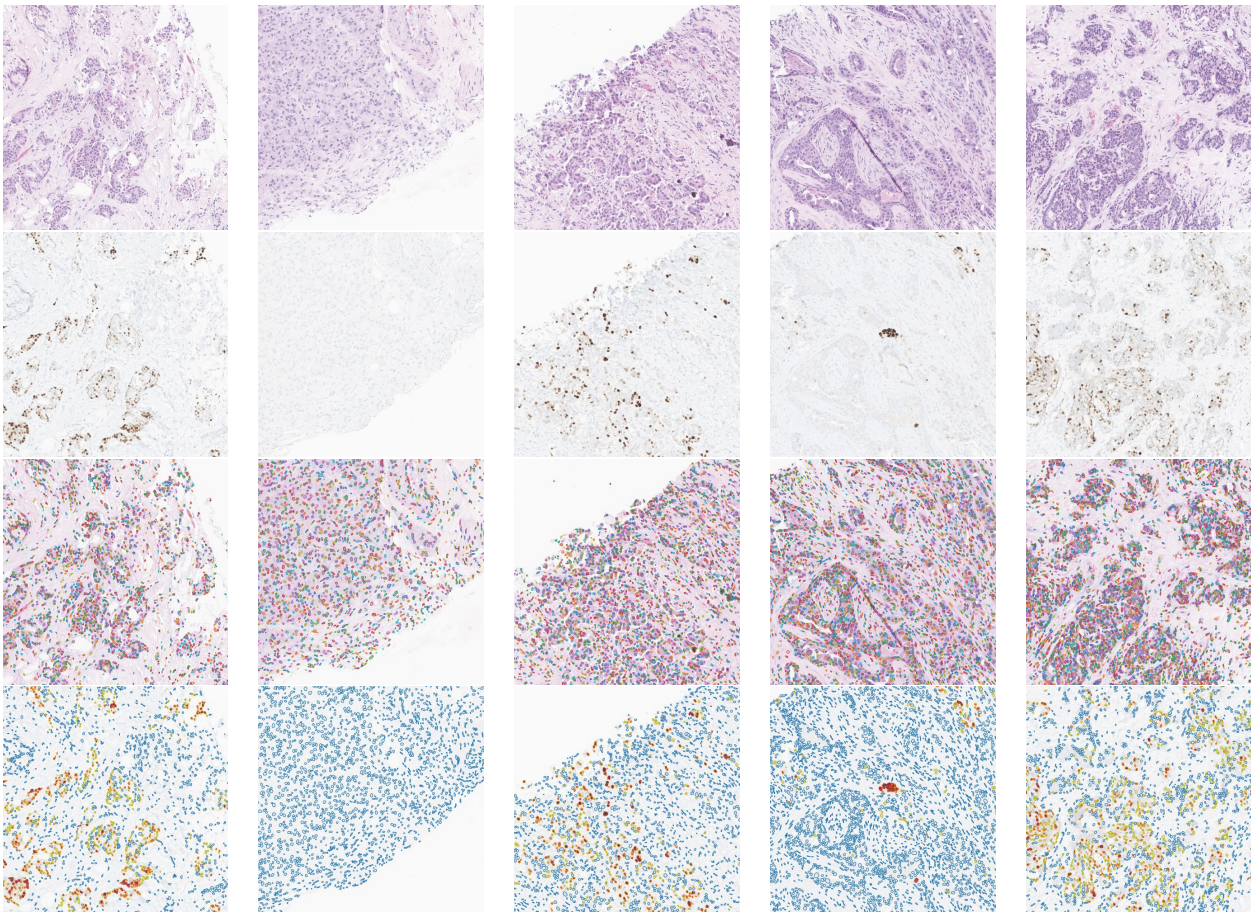


Figure 5.4: Examples of H&E-PR pairs in the dataset. Each column corresponds to a H&E-IHC pair. Row1: H&E, Row2: IHC, Row3:nuclei sementations on H&E, Row4: the result of DAB-analysis on IHC where blue, yellow, orange, and red colors mark 0 (*i.e.* negative), 1+ (*i.e.* weakly stained), 2+ (*i.e.* moderately stained), and 3+ (*i.e.* strongly stained) nuclei, respectively.

corresponding H&E patch. We used StarDist [118] to segment the nuclei in the H-DAB patch. Afterwards, we used the conventional color-deconvolution to extract the brown DAB channel [117]. In the dataset for each IHC image we have included the average DAB channel within every and each nucleus, so different numbers like H-Score, percentage, and Allred score can be computed for each H&E-IHC pair. The total number of nuclei (i.e. the denominator in percentage or H-score) was obtained from H&E images, since nuclear segmentation was found more reliable when using StarDist [118] on the hematoxylin stain of the H&E assay compared to the hematoxylin of the H-DAB assay. Indeed, we noticed that in the H-DAB modality StarDist [118] may miss many negative nuclei or may mistakenly take artifacts as negative nuclei. Some examples are provided in Fig. 5.3 and its caption. Therefore, in the dataset, we include the total number of nuclei detected in each H&E patch, which is a more reliable estimate for the total number of nuclei when computing percentage (for Ki-67) or H-score (for ER and PR). Figs. 5.2, 5.3, and 5.4 show H&E patches (row 1), the

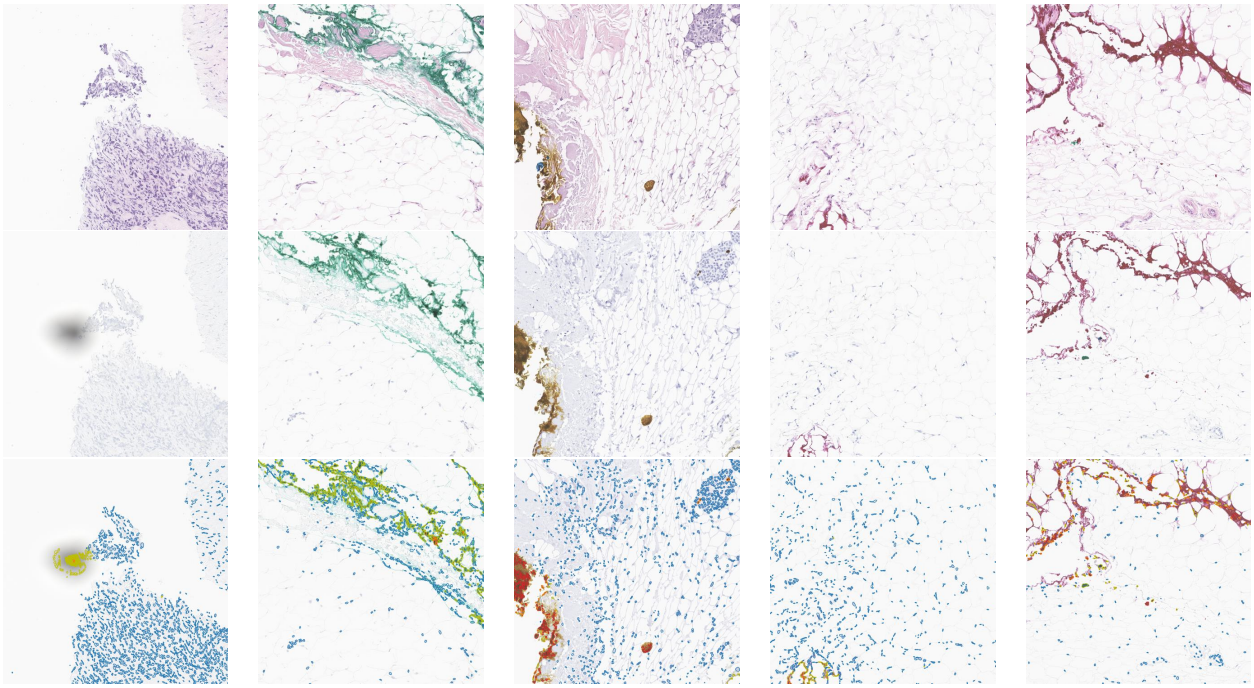


Figure 5.5: Examples of the failure of DAB-analysis, discussed in Sec. 5.3.4. Each column corresponds to a H&E-IHC pair. Row1:H&E, Row2: IHC, Row3: The result of DAB-analysis where blue, yellow, orange, and red colors mark 0, 1+, 2+, and 3+ nuclei.

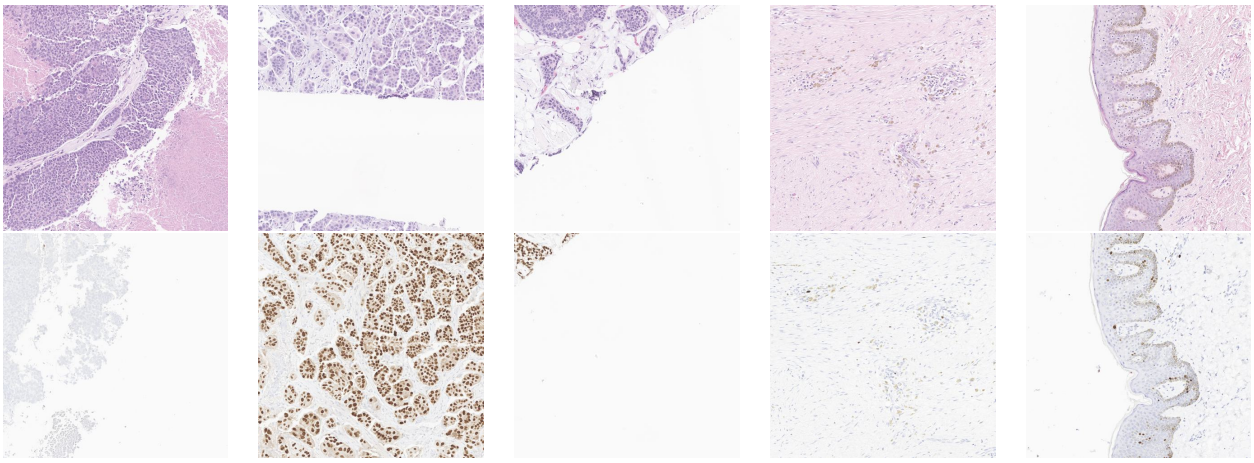


Figure 5.6: Examples of H&E-IHC pairs which are discarded during the exhaustive visual inspection. Each column corresponds to a H&E-IHC pair. Row1: H&E, Row2: IHC. Details are provided in Sec. 5.3.4

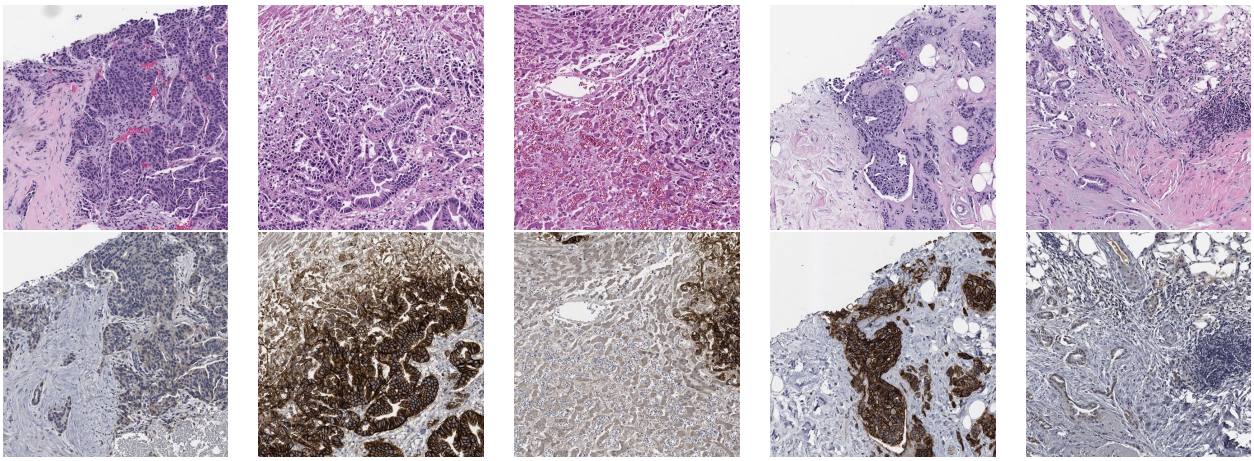


Figure 5.7: Examples of HER2 pairs in the dataset. Each column corresponds to a H&E-IHC pair. Row1: H&E, Row2: IHC. In columns 2, 3, and 4 some 3+ patterns exist in the H-DAB modality and the corresponding regions exist in the H&E modality. So those pairs are labeled as positive during the exhaustive inspection. On the other hand, the pairs in the 1st and 5th columns are labeled as negative.

corresponding IHC patch (row 2), the result of nuclei segmentation on H&E (row 3), and the result of H-DAB analysis (row 4) for Ki67, ER, and PR, respectively. The H-score has a range of 0 to 300 and is computed as follows: (% of weakly stained nuclei * 100) + (% of moderately stained nuclei * 200) + (% of strongly stained nuclei * 300). The determination of weak, moderate, and strong intensities was based on two fixed thresholds for DAB optical density mean established by one pathologist. A single (weak) threshold was established for the Ki67 percentage assessment. In this study, nuclei were not discriminated between tumoral and non-tumoral for H-score calculation.

5.3.4 Visual Inspection of Pairs and DAB-Analysis

After obtaining corresponding H&E and H-DAB mirrored images and corresponding analytical labels, not all pairs are valid due to several reasons. So an expert pathologist (co-supervisor) worked closely with a non-expert (the writer of this thesis) to exclude all invalid pairs from the dataset. A total of 147404 pairs were exhaustively inspected, out of which 54635 pairs were discarded and 22734 pairs were manually labeled as zero (as we discuss below, some instances like 1^{st} and 2^{nd} columns of Fig. 5.5 were manually labeled as zero).

Fig. 5.5 illustrates problematic images which provide wrong H-DAB analysis results. Each column illustrates one example. The rows depict the H&E patch images, H-DAB patch images, and DAB-analysis images performed on H-DAB, respectively. First column: out of focus non-tissular artifact; second column green ink; third column yellow ink; and columns 3-5 red/brown inks. These non-tissular events are mistakenly considered as positive signal by the automatic DAB-analysis. In

the case of 1st and 2nd columns, apart from the wrong positive signal there are no real positive nuclei elsewhere. So, in such cases, mirrored images are kept in the dataset but a manual label (H-Score) of 0 was assigned to the pair. Keeping these images (with proper H-Score) present opportunities for machine learning model to be exposed to such artifacts. On the other hand, in cases like the one in the 3rd column, besides the false positive signals, there are some real positive nuclei in the two o'clock position which made it difficult to ensure a proper H-score label. Such cases were discarded from the dataset.

Another reason for discarding a pair is imperfect registration (column 3 of Fig. 5.6) or some tissue parts being missing in the corresponding modality (columns 1 and 2 of Fig. 5.6). These cases were discarded from the dataset during the exhaustive visual inspection. Other pairs were removed, as illustrated in the 4th and 5th columns of Fig. 5.6, when they contain brown pigments like hemosiderin and melanin identified in both modalities (H&E and IHC), producing false DAB signal in the corresponding IHC image. During the visual inspection, if there were such signals in the H&E image, the pair was discarded. For such WSIs not all patches (only the ones with brown color in H&E) were excluded, to make sure that these cases are not completely discarded from the dataset. Most patches from the white slide background were discarded. For each region a few white patches (containing no tissue) were included in the dataset so a machine learning model can learn to label the white background as negative.

5.3.5 Obtaining Labels for HER2 Pairs

HER2 protein is a membrane-bound receptor and therefore the corresponding DAB signal is membranous. Thus for HER2 IHC, as a first attempt we chose not to include the 2+ cases due to poor inter-observer reproducibility [108] for 2+ cases. All H&E-HER2 pairs were inspected one-by-one. A pair was labeled as positive if any 3+ patterns were present in the IHC modality and the corresponding regions were present in the H&E modality. In other words, a binary label was assigned to each H&E-HER2 pair. During the exhaustive inspection, some pairs were discarded due to the reasons that we discussed in Sec. 5.3.4. For each WSI we had access to pathologist assessment result as 0, 1+, 2+, or 3+. In this study we did not include equivocal (2+) cases but instead focused on a binary approach: WSIs labeled as 0 or 1+ (high probability of non-amplified status) and WSIs labeled as 3+ (high probability of amplified status). We exhaustively inspected the pairs to confirm the veracity of this binary labeling between HER2 positive and negative cases. Some example patch pairs are provided in Fig. 5.7. For example, for the pairs in columns 2, 3, and 4 of Fig. 5.7 some 3+ patterns exist in the IHC and the corresponding regions exist in the H&E modality. So those pairs are labeled as positive, while the pairs in the 1st and 5th columns are labeled as negative.

5.3.6 Dataset Statistics

For ER, PR, Ki67, and HER2 59, 60, 60, and 52 (total of 231) WSI pairs were used, respectively. For ER, PR, Ki67, and HER2 41098, 38914, 31631, and 41098 patch pairs were obtained, respectively, according to the procedure of Sec. 5.3.2. Among the extracted patch-pairs, for ER, PR, Ki67, and HER2 10703, 13943, 9886, and 20103 patch pairs were discarded and 4122, 2872, 1756, and 13984 were manually set to 0 during the exhaustive inspection, respectively. Some dataset samples are illustrated in Figs. 5.2, 5.3, 5.4, and 5.7.

This work and the authorization to publish the anonymized dataset in the public domain https://ihc4bc.github.io/, received the ethical approval (HREBA.CC-19-0347) from the Health Ethics Board of Alberta.

5.4 Experiments: Training with 3K-by-3K-Level Labels

5.4.1 Methods

We used the state-of-the-art WSI classification method Clustering-constrained Attention Multiple Instance Learning (CLAM) [89] whenever experimenting with weak WSI-level labels. When dealing with patch-level labels, we firstly resized each 3K by 3K patch to 1K by 1K pixels. Afterwards, we used a pipeline that processes a 1K by 1K patch as follows. A 1K by 1K patch is divided to a 4 by 4 grid whose cells are 250 by 250 pixels. Consequently, each grid cell is fed to a Resnet-18 [60] backbone to get a volumetric map of shape $512 \times 8 \times 8$ for each cell of the grid. These volumetric maps are concatenated to form a volumetric map of shape $512 \times 32 \times 32$, which is fed to a multi-head self-attention layer (used in vision transformers [53]) followed by a classification head with a linear layer. We used a publicly available implementation of vision transformers (ViT) [53] in PyTorch [8]. During training we used the following data augmentations: $\pm 10\%$ jitter to saturation and hue channels, and ± 180 degrees random rotation to images.

5.4.2 Labeling Protocols for Classifiers

In this study we did not use regression to predict the continuous numbers (i.e. H-scores for ER, PR, and percentage for Ki67). Instead, we used a simple method known as cumulative logits approach [29] which puts a set of thresholds like T_1, T_2, ..., T_M on the continuous value where $T_1 < T_2 < ... < T_M$. Afterwards, for each threshold a separate classifier is considered; classifier 1 predicts whether the continuous value is below T_1 or above T_1, classifier 2 predicts whether the continuous value is below T_2 or above T_2, ..., classifier M predicts whether the continuous value is below T_M or above T_M. Using this approach comes with the following issue. Let ϵ be a small

number. The m-th classifier is supposed to assign a sample with number $T_m - \epsilon$ and a sample with number $T_m + \epsilon$ to different classes. In other words, the gray-zone samples whose number is close to T_m may confuse the m-th classifier. To avoid this issue, we also tested a slightly modified approach as follows. Classifier 1 predicts whether the continuous value is below T_1 or above T_2, classifier 2 predicts whether the continuous value is below T_2 or above T_3, ..., classifier $M-1$ predicts whether the continuous value is below T_{M-1} or above T_M. We refer to the former and the latter approaches as "without gray-zone" and "with gray-zone", respectively.

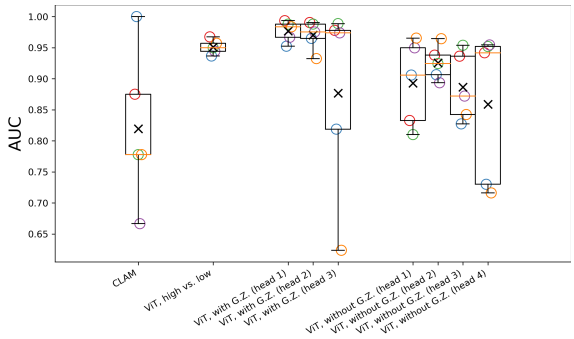
5.4.3 Training Setup

For the weakly-supervised method CLAM [89] we used the default parameter settings available in the public repository [9]. When training the strongly supervised patch classifiers, we always used an AdamW [88] optimizer with the AMSGrad [110] flag enabled. We trained with two learning rates, 0.0001 and 0.00001 and picked the checkpoint with the highest validation AUC. To have a fair comparison, the training cases shown to the CLAM [89] and the patch classifiers should be as close as possible. To this end, we firstly select 90 percent of WSIs for training the WSI-classifier CLAM [89] and the rest for testing. Consequently, the patches from the training/testing whole-slide images are used as training/testing sets for the patch classifiers. By doing so, the WSI-classifier CLAM [89] and the patch classifier are provided with the same set of training cases. To make a validation set for a patch classifier, for each WSI in the training set we randomly selected one of the annotated regions (like the regions shown in the third column of Fig. 5.1) and considered its patches in the validation set. For CLAM [89] we randomly selected 5 WSIs as the validation set. Note that due to class imbalance the random testing/validation splits may become devoid of instances from some classes. In this study we did not use such splits in the experiments.

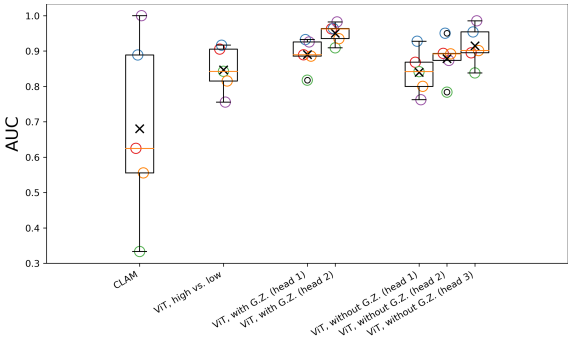
5.4.4 Results

Prediction Performances

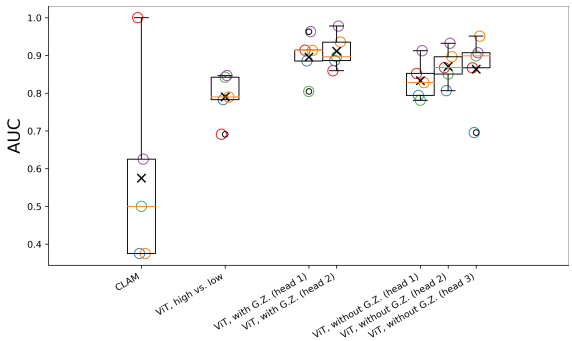
For each marker we created 5 splits according to the procedure of Sec. 5.4.3. When presenting the results, each split has its own unique color. For example in Fig. 5.8a the red circles correspond to a fixed training/testing split which is used by different methods and labeling protocols. Prediction performances in terms of AUC are provided in Figs. 5.8a, 5.8b, 5.8c, and 5.8d. In these figures the first box-plot on the left shows the performance of CLAM [89] in classifying WSIs to two classes. For HER2 (Fig. 5.8d the box-plot on the left) WSIs from 0 and 1+ cases are labeled as CLAM [89]'s class 0 and WSIs from 3+ cases are labeled as CLAM [89]'s class 1. For Ki67, ER, and PR we firstly computed the median of the continuous values in the dataset, which are 3.82 for Ki67 percentage, 42.61 for ER H-score, and 7.373 for PR H-score. Afterwards, we used these median



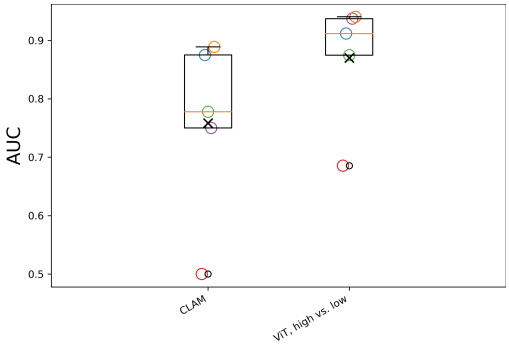
(a) Results for predicting Ki67 status. From left to right, 1st box-plot: CLAM [89] when predicting WSI-level Ki67-percentage below 3.82 versus above 3.82. 2nd box-plot: patch-level Ki67-percentage below 3.82 versus above 3.82. 3rd-5th box-plots: patch-level Ki67-percentage (head 1: below 5 versus above 10, head 2: below 10 versus above 15, head 3: below 15 versus above 20). 6th-9th box-plots: patch-level Ki67-percentage (head 1: below 5 versus above 5, head 2: below 10 versus above 10, head 3: below 15 versus above 15, and head 4: below 20 versus above 20).



(b) Results for predicting ER status. From left to right, 1st box-plot: CLAM [89] when predicting WSI-level ER H-score below 42.61 versus above 42.61. 2nd box-plot: patch-level ER H-score below 42.61 versus above 42.61. 3rd-4th box-plots: patch-level ER H-score (head 1: below 30 versus above 60, head 2: below 60 versus above 90). 5th-7th box-plots: patch-level ER-percentage (head 1: below 30 versus above 30, head 2: below 60 versus above 60, head 3: below 90 versus above 90).



(c) Results for predicting PR status. From left to right, 1st box-plot: CLAM [89] when predicting WSI-level PR H-score below 7.373 versus above 7.373. 2nd box-plot: patch-level PR H-score below 7.373 versus above 7.373. 3rd-4th box-plots: patch-level PR H-score (head 1: below 30 versus above 60, head 2: below 60 versus above 90). 5th-7th box-plots: patch-level PR-percentage (head 1: below 30 versus above 30, head 2: below 60 versus above 60, head 3: below 90 versus above 90).



(d) Results for predicting HER2 status. From left to right, 1st box-plot: CLAM [89] when predicting WSI-level HER2 status positive versus negative. 2nd box-plot: predicting whether 3+ patterns exist in a patch (positive/high) or not (negative/low).

Figure 5.8: Prediction performances for a) Ki67, b) ER, c) PR, and d) Her2 in terms of ROC-AUC.

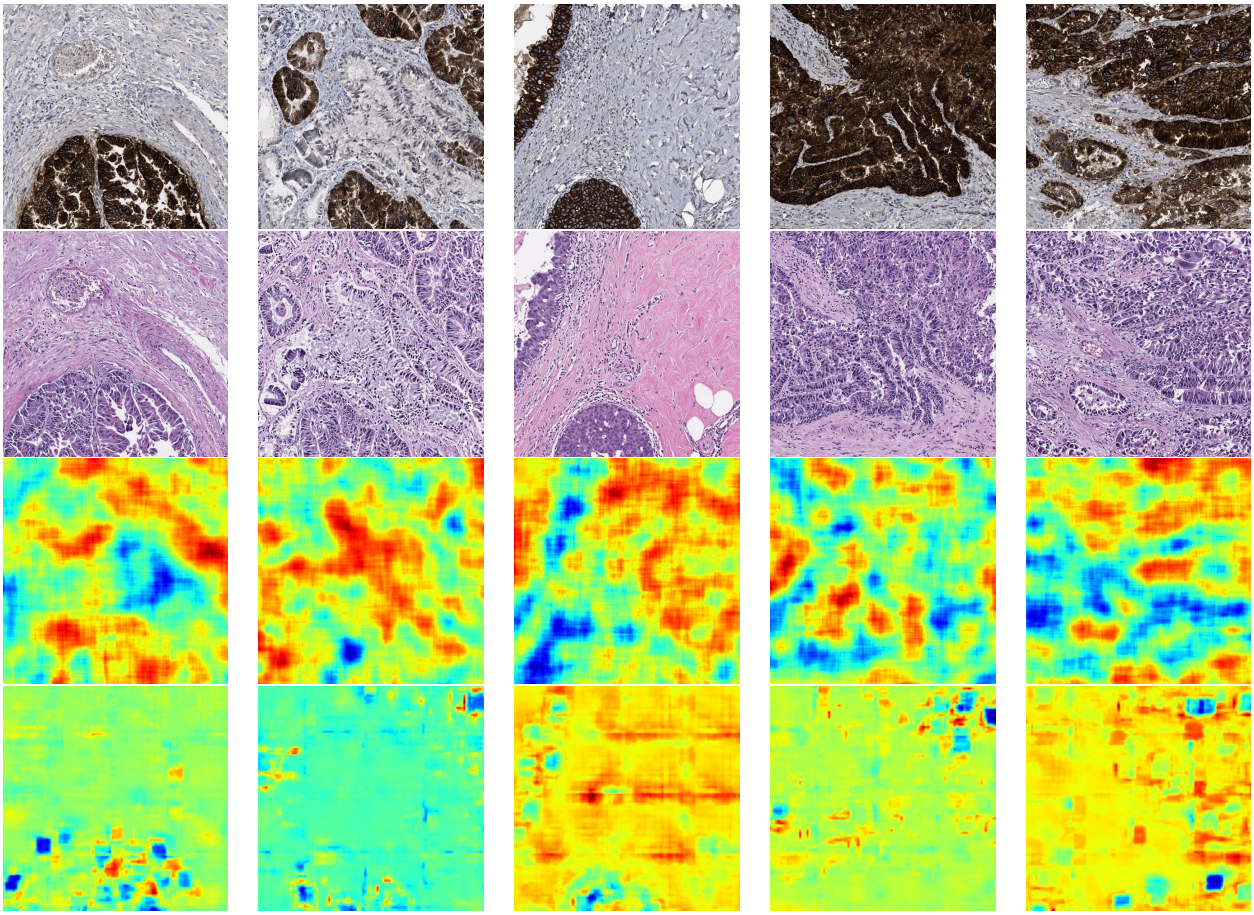


Figure 5.9: Localizations for HER2 predictors trained/evaluated on the split shown by the orange circle in Fig. 5.8d. Each column corresponds to a H&E-IHC pair. Row 1: IHC, row 2: H&E, row 3: CLAM [89]'s attention mask, row 4: patch classifier's sensitivity to pixels. In all heatmaps we used JET color-map in which high and low values appear in red and blue, respectively.

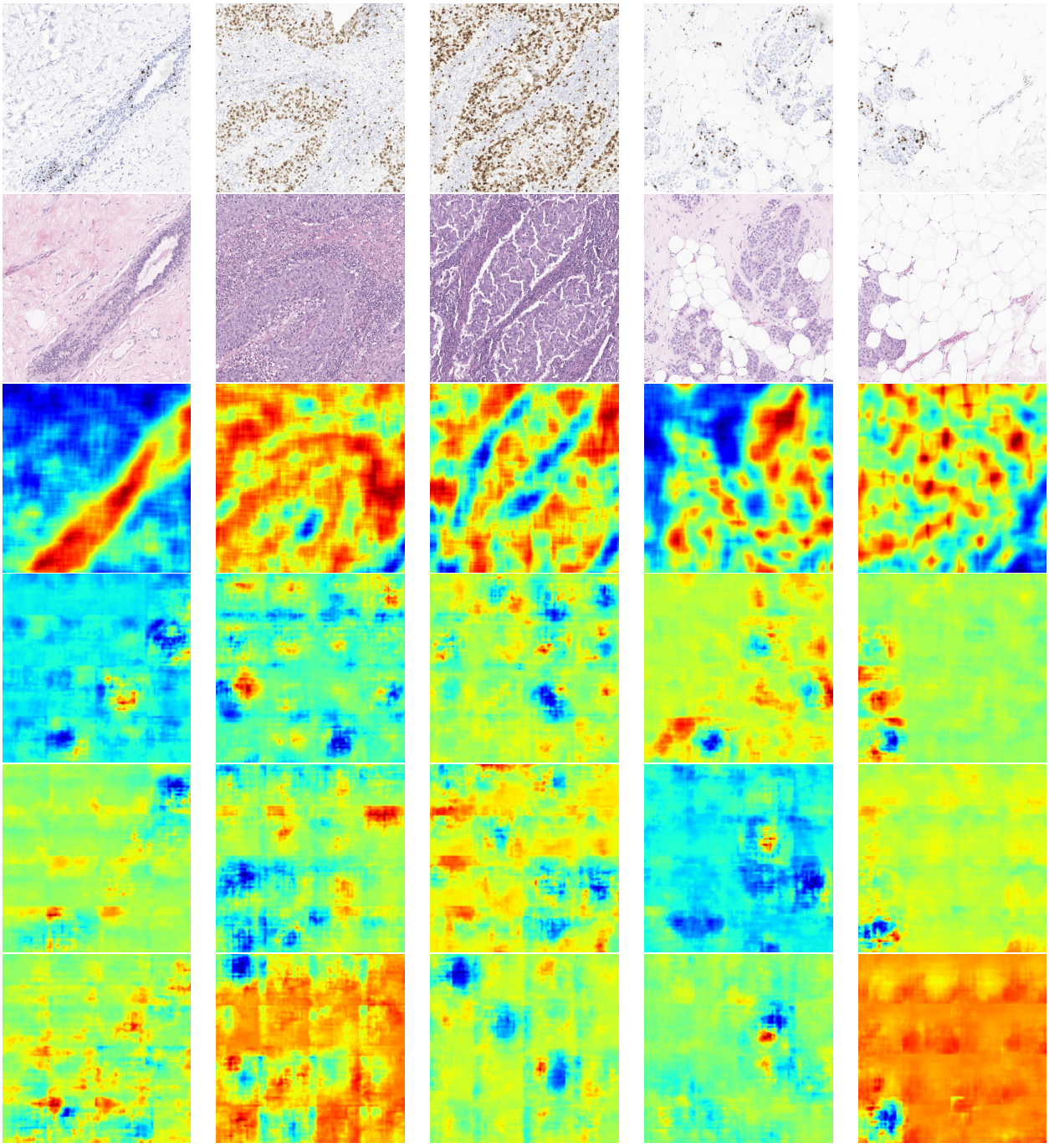


Figure 5.10: Localizations for Ki67 predictors trained/evaluated on the split shown by the red circle in Fig. 5.8a. Each column corresponds to a H&E-IHC pair. Row 1: IHC, row 2: H&E, row 3: CLAM [89]'s attention mask, row 4: the sensitivity of the classifier labeled as "ViT, high vs. low". rows 5: the average sensitivity of heads of the classifier labeled as "ViT, with G.Z.". rows 6: the average sensitivity of heads of the classifier labeled as "ViT, without G.Z.". In all heatmaps we used JET color-map in which high and low values appear in red and blue, respectively.

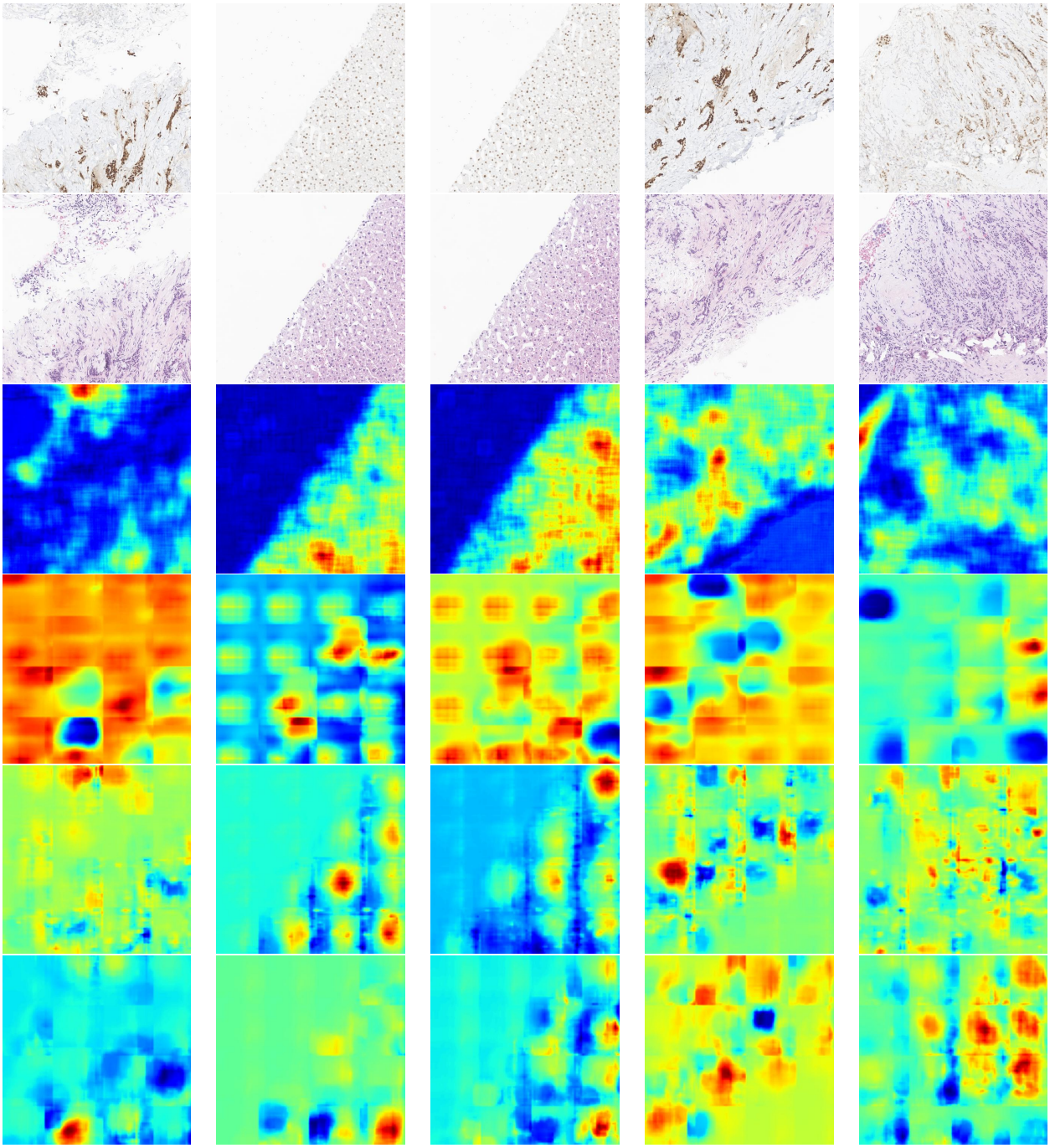


Figure 5.11: Localizations for ER predictors trained/evaluated on the split shown by the purple circle in Fig. 5.8b. Each column corresponds to a H&E-IHC pair. Row 1: IHC, row 2: H&E, row 3: CLAM [89]'s attention mask, row 4: the sensitivity of the classifier labeled as "ViT, high vs. low". rows 5: the average sensitivity of heads of the classifier labeled as "ViT, with G.Z.". rows 6: the average sensitivity of heads of the classifier labeled as "ViT, without G.Z.". In all heatmaps we used JET color-map in which high and low values appear in red and blue, respectively.

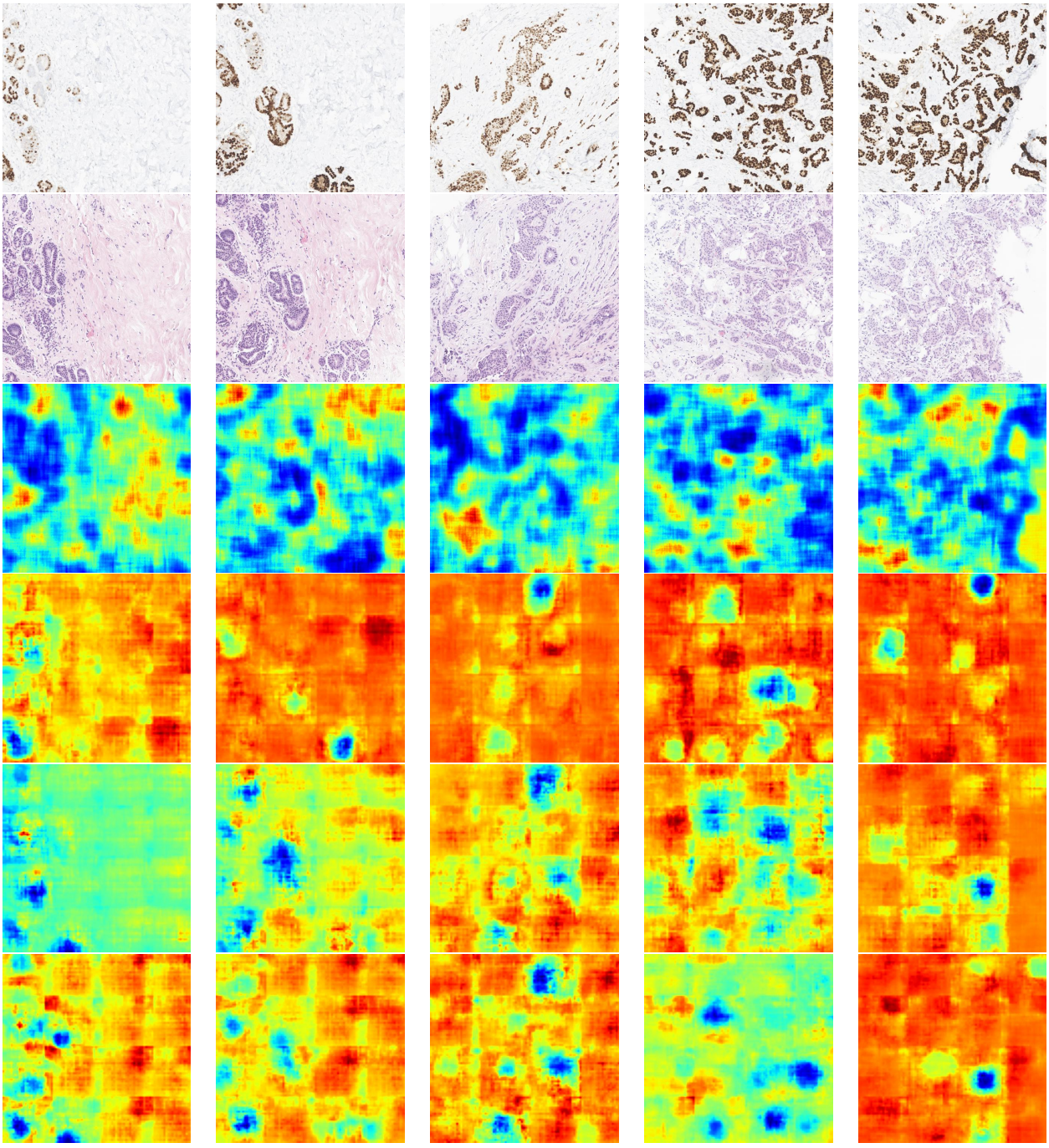


Figure 5.12: Localizations for PR predictors trained/evaluated on the split shown by the red circle in Fig. 5.8c. Each column corresponds to a H&E-IHC pair. Row 1: IHC, row 2: H&E, row 3: CLAM [89]'s attention mask, row 4: the sensitivity of the classifier labeled as "ViT, high vs. low". rows 5: the average sensitivity of heads of the classifier labeled as "ViT, with G.Z.". rows 6: the average sensitivity of heads of the classifier labeled as "ViT, without G.Z.". In all heatmaps we used JET color-map in which high and low values appear in red and blue, respectively.

values to make WSI-level labels for CLAM [89]; if the WSI-level continuous number is below the median value it belongs to class 0 and otherwise to class 1. We evaluated the performance of the vision transformer-based (ViT) [53] patch classifier with the same thresholds used for CLAM [89]. The results are provided in Figs. 5.8a, 5.8b, 5.8c, and 5.8d by the box-plots labeled as "ViT, high vs. low". Note that, for example in Fig. 5.8a CLAM [89]'s task is to predict whether the WSI-level Ki67 percentage is below or above 3.82. But the patch classifier (the box-plot labeled "ViT, high vs. low") predicts whether the patch-level (as opposed to WSI-level) Ki67 percentage is below or above 3.82. We also experimented with the cumulative logits approach, as explained in Sec. 5.4.2 and in two settings: with gray-zone and without gray-zone. In Figs. 5.8a, 5.8b, and 5.8c the corresponding box-plots are labeled as "ViT with G.Z." and "ViT without G.Z.", respectively. We used the following thresholds on Ki67 percentage: 5, 10, 15, and 20. Moreover, for ER and PR we used the following thresholds on H-score: 30, 60, and 90.

Localizations

Besides reporting the prediction performances, we inspected to what degree the weakly-supervised method CLAM [89] and the ViT-based pipeline described in Sec. 5.4.1 can localize relevant regions. CLAM [89] has an explicit attention mechanism, so for CLAM [89] we traversed a 1K by 1K image with a sliding window of size 256 and stride of 10, feeding each 256 by 256 patch to CLAM [89]'s attention sub-module to obtain a heatmap. If the heatmap has a large value at a pixel position, intuitively CLAM [89] has payed more attention to that pixel position. The ViT-based pipeline has no explicit attention mechanism, so to highlight the important pixel positions we used an attribution-based approach as follows. A 1K by 1K patch is traversed with a white patch of size 50 and stride 10, and the average change in pipeline's output is recorded in the heatmap. Intuitively, the sliding white patch hides a small region of the input image to see how it affects the pipeline's output. Note that there are more sophisticated feature-attribution methods, but they may produce discordant explanations [78]. In all heatmaps, we used JET color-map in which high and low values appear in red and blue, respectively. The localizations for HER2, Ki67, ER, and PR are provided in Figs. 5.9, 5.10, 5.11, and 5.12.

5.4.5 Discussion and Analysis

The performance of the WSI classification method CLAM [89] in Figs. 5.8a, 5.8b, 5.8c, and 5.8d (the 1st bars from left) shows that training with WSI-level labels is not effective for predicting ER, PR, Ki67, and HER2 statuses. This observation is consistent with the study done by Laleh *et al.* [84]. The best results correspond to the ViT-based pipeline trained on 3K-by-3K labels and with the gray-zones explained in Sec. 5.4.2 (labeled as "ViT-with G.Z." in Figs. 5.8a, 5.8b, 5.8c, and labeled

as "ViT, high vs. low" in Fig. 5.8d). In this setting the prediction performances approaches or exceeds 90 in terms of AUCROC. Interestingly, when the gray-zone is not considered (labels as "ViT, without G.Z." in Figs. 5.8a, 5.8b, 5.8c, and 5.8d) we see a drastic drop in prediction performances. This shows that when the gray-zones are not considered, the small flaw of assigning different labels to patches with H-scores $T_m - \epsilon$ and $T_m + \epsilon$ (i.e. the issue elaborated upon in Sec. 5.4.2) can significantly reduce the prediction performance. In our best results (labeled as "ViT-with G.Z." in Figs. 5.8a, 5.8b, 5.8c, and labeled as "ViT, high vs. low" in Fig. 5.8d) the prediction performances approach or exceed 90% in terms of AUC for most splits, but we see a drop of performance for some splits. Since roughly 240 WSIs are used to create the dataset, we hypothesize that in those splits some tissue-types in the test set happen to be missing in the training set. In other words, one might still need to expand the dataset to expose machine learning models to more morphological variety during training.

Besides reporting the prediction performances, we obtained the localization maps from different methods according to the procedure of Sec. "Localizations". These localization maps are provided in Figs. 5.9, 5.10, 5.11, and 5.12 for HER2, Ki67, ER, and PR, respectively. In these figures the five columns show different (H&E)-(H-DAB) examples: the 1^{st} and 2^{nd} rows depict respectively the H&E and H-DAB images. Brown regions in an H-DAB image correspond to positive regions and the corresponding regions in the H&E modality should ideally be highlighted by machine learning methods. The 3^{rd} and 4^{th} rows respectively illustrate the CLAM [89]'s and the ViT-based pipeline's heatmaps. According to these heatmaps, although both pipelines have achieved around 90 AUCs neither of them can successfully localize the relevant regions. For example, according to the heatmap in 3^{rd} column and 3^{rd} row of Fig. 5.9, CLAM [89] mistakenly pays attention to the negative region in the two o'clock position. This observation is consistent with the experiments done by Laleh *et al.* [84] in which weakly-supervised methods mistakenly highlight tissue borders or other artefact when trained to predict molecular information. Another observation is that the heatmaps related to the ViT-based pipeline (i.e. the heatmaps in the 4^{th} row) are often homogeneous. This might be due to the fact that the pipeline is by-design good at making use of contextual information. More precisely, although a CNN normally takes in or "sees" a relatively small patch, the ViT-based pipeline explained in Sec. 5.4.1 can see a 1K by 1K patch in its entirety. All in all, although in Figs. 5.9, 5.10, 5.11, and 5.12 the selected checkpoints can achieve around or above 90% AUCs, none of them can successfully localize relevant tissue regions. Intuitively, during training the pipelines are never asked to decide if each region is positive or negative, and they merely need to output a label for a patch or WSI by, *e.g.*, detecting only a small portion of relevant regions.

5.5 Experiments: Pixel-level Supervision versus Automatic Localization

5.5.1 Acquiring Pixel-level Labels

Near 900K (exactly 881,957) spots were manually annotated on HER2-positive image regions of the IHC4BC dataset. Fig. 5.13 illustrates sample spots which are manually placed on H&E images by looking at the corresponding positive regions in the H-DAB modality.

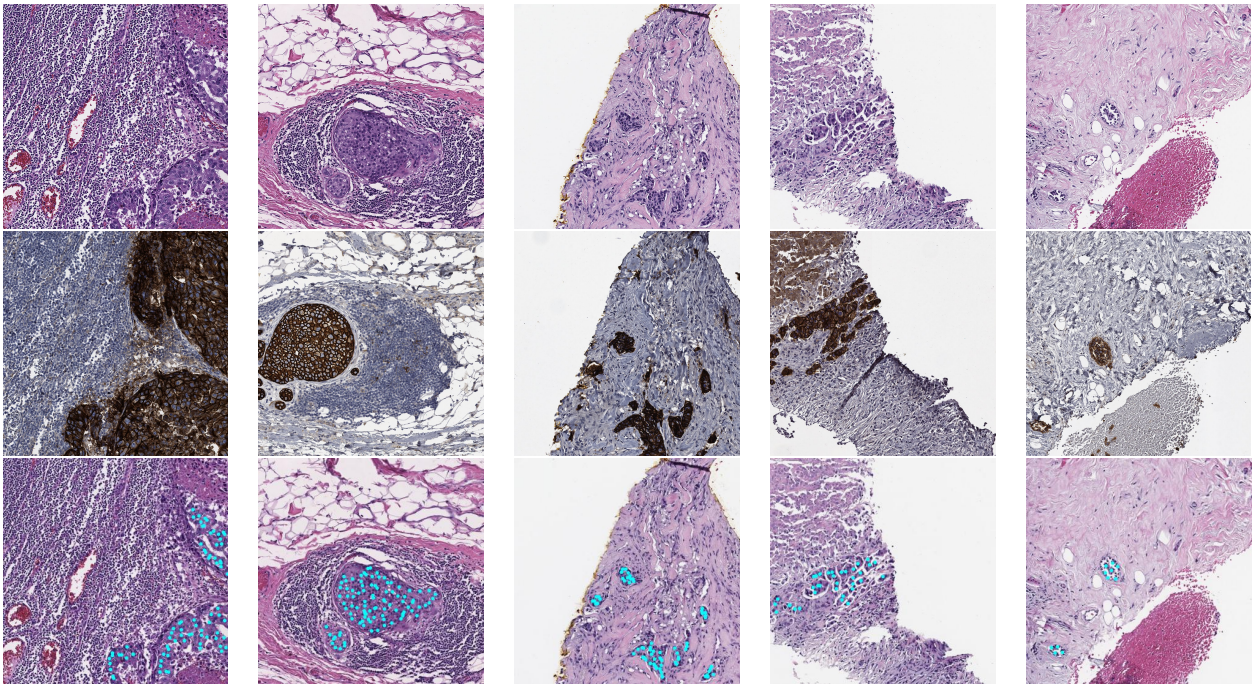


Figure 5.13: Examples of manual annotations on Her2 positive regions. Each column corresponds to a H&E-IHC pair. Row1: H&E, Row2: H-DAB, Row3: the cyan spots show the manually marked Her2 positive regions.

5.5.2 Experimental Setup

The research question is: can a state-of-the-art automatic localization method with 3K by 3K level labels be as good as a strongly-supervised classifier trained with the manual pixel-level spots? To answer this question we benchmarked two methods

- 1. A simple classifier with Resnet-18 [60] backbone trained on 120 by 120 patches and with 120 by 120 level labels.

- 2. Co-teaching [58], a sophisticated automatic localization method trained on 120 by 120 patches where the 3K by 3K level labels are assigned to each 120 by 120 patch.

For the simple classifier the 120 by 120 positive patches were extracted by first randomly selecting a positive spot, then adding ± 25 pixels uniform random shift to the pixel location and then extracting a 120 by 120 patch from that pixel location near the manually marked spot. For both methods we used color jitter with values 0.2, 0.0, 0.1, and 0.1 for brightness, contrast, saturation, and hue, respectively. We used Adam optimizer [80] with learning rates 0.0001 and 0.001 for the simple classifier and co-teaching, respectively. An important hyper-parameter of co-teaching [58] is an estimate for the noise-rate, which in our setting is equal to the percentage of HER2 negative 120 by 120 patches mistakenly labeled as positive (*i.e.* negative patches extracted from HER2 positive 3K by 3K images). To estimate this hyper-parameter, we applied DAB-deconvolution [117] to HER2 positive 3K by 3K images and used the DAB channel to estimate the percentage of HER2 negative pixels over all 3K by 3K HER2 positive images. This estimate turned out to be 0.45. To have a solid evaluation, we used leave-one-patient-out evaluation where at each round images from a single patient are withheld for evaluation. To make a validation set - similar to the setup of Sec.5.4.3 - for each WSI in the training set we randomly selected one of the annotated regions (like the regions shown in the third column of Fig. 5.1) and considered its patches in the validation set. The checkpoint with the highest validation performance in terms of AUC is picked as the best checkpoint. AUC is not usable in the leave-one-patient-out setting, because the left out patient may not have any HER2 positive images in which case the AUC is not defined. Therefore we used relevance score, which is simply the predicted probability of the correct class. Note that since we have ground-truth manual spots, we reported the relevance score over different 120 by 120 patches from images of test patients. Also note that negative 120 by 120 patches are only extracted from HER2 negative 3K by 3K patches, because some positive regions of HER2 positive 3K by 3K images may have no manual spot on them.

5.5.3 Results

Fig. 5.14 shows the truncated violin plots for the leave-one-patient-out analysis where the caption of each subfigure specifies the anonymous identifier of the patient who is held out for testing (*e.g.* "patient 1009", "patient 3009", etc.) The very last subfigure of Fig. 5.14 with caption "Overall (all patients)" depicts the violin plot for all relevance scores. Fig. 5.15 illustrates prediction probability heatmap of the strongly-supervised classifier (columns 3 and 4) and that of co-teaching [58] (columns 5 and 6) for six H&E-H-DAB image pairs. Rows belong to the following patient identifiers which can be matched to patient identifiers in subfigure captions of Fig. 5.14: 66009, 78009, 501009, 19009, 150009, and 233009.

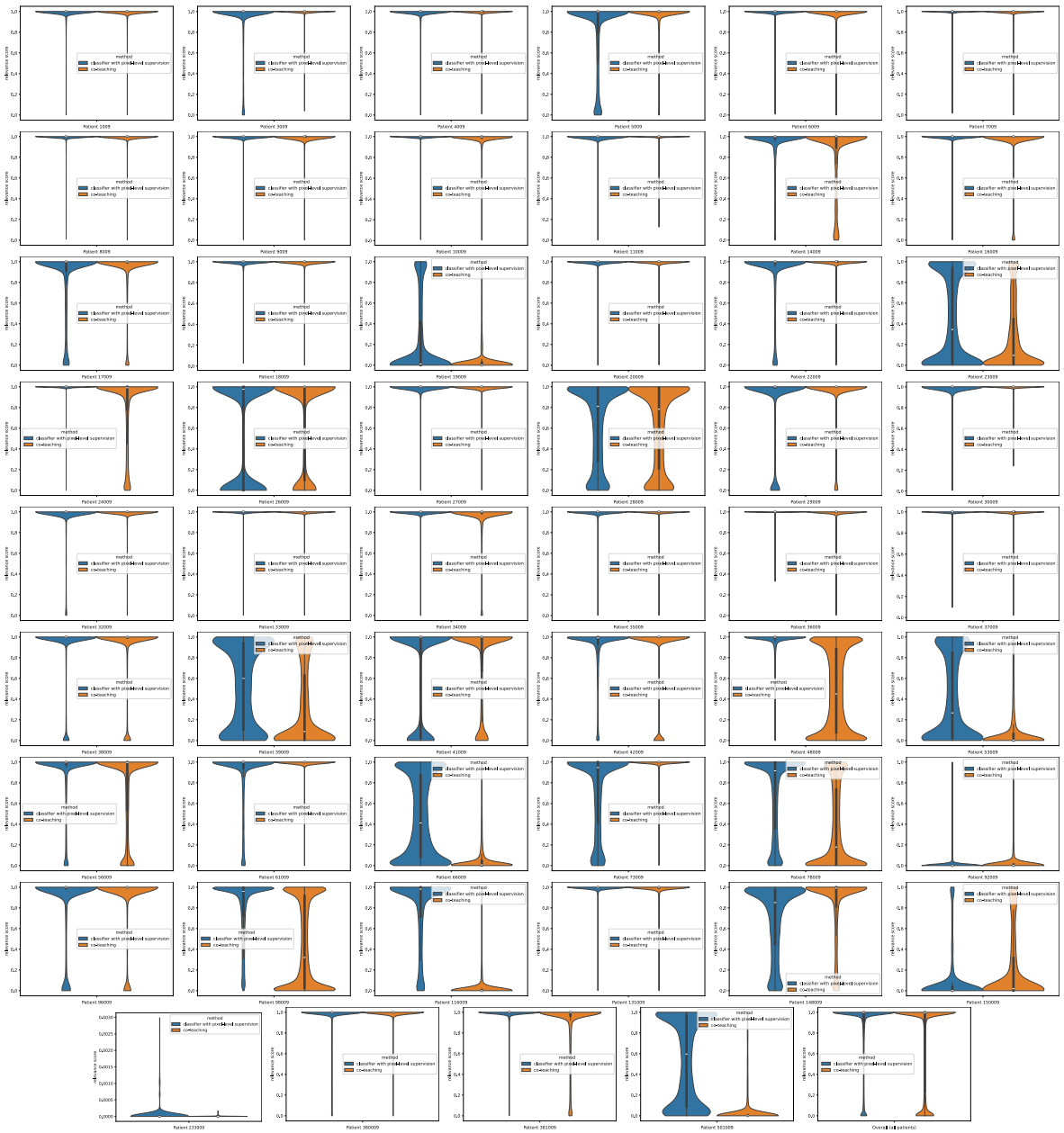


Figure 5.14: Leave-one-patient-out evaluation in terms of relevance score.

5.5.4 Discussion and Analysis

According to Fig. 5.14 co-teaching [58] is competitive with the classifier trained with pixel-level labels, although it only has access to 3K by 3K level labels. Indeed, although in Sec. 5.4.4 we showed that for the problem at hand automatic localization is not possible with WSI-level labels, but in Fig. 5.14 we observe that doing so is possible with 3K by 3K level labels. Recall that WSIs are huge (*e.g.* 100K by 100K pixels in size), and localization with WSI-level labels is similar to

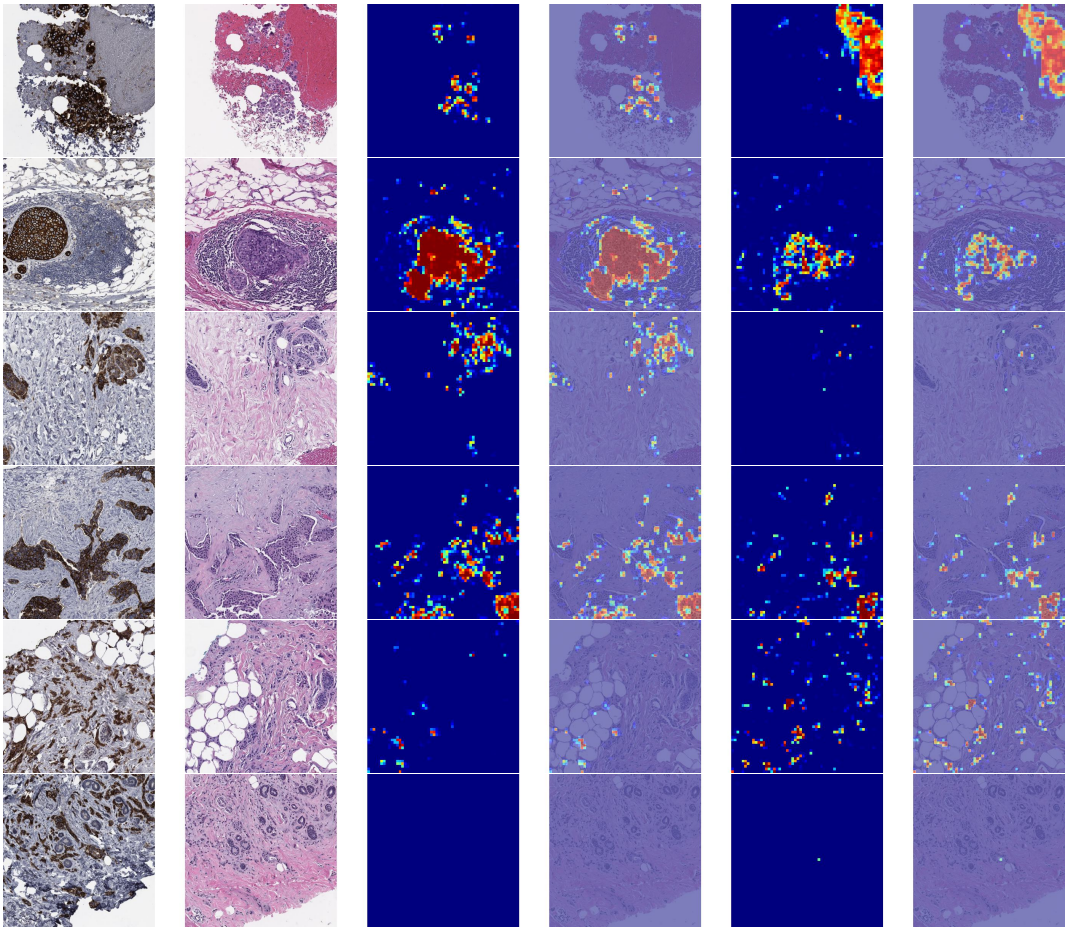


Figure 5.15: Sample heatmaps of strongly-supervised classifier and co-teaching [58]. Heatmaps are shown with the jet colormap with minimum (resp. maximum) value of 0.0 (resp. 1.0). column 1: H-DAB image, column 2: H&E image, column 3: simple classifier's activation map, column 4: same as column 3 but overlayed on the H&E image, column 5: activation map from co-teaching, column 6: same as column 5 but overlayed on the H&E image. Rows belong to the following patient identifiers which can be matched to patient identifiers in Fig. 5.14: 66009, 78009, 501009, 19009, 150009, and 233009.

"finding a needle in a haystack". But a 3K by 3K patch is a smaller search space for localization, and that may explain the success of co-teaching in Fig. 5.14.

In Fig. 5.14 for the majority of patients both methods have performed well. There are cases where co-teaching has failed but the strongly-supervised classifier has not (*e.g.* "Patient 116009" in the second last row and "Patient 501009" in the last row of Fig. 5.14). Moreover, there are cases where both methods have failed (*e.g.* "Patient 233009" in the last row of Fig. 5.14). Intriguingly, there are cases where co-teaching [58] has performed slightly better than the strongly-supervised classifier (*e.g.* "Patient 150009" in the second last row of Fig. 5.14). The heatmaps of Fig. 5.15 provides more insights.

- The first row of Fig. 5.15: the strongly-supervised classifier has been quite successful (columns 3 and 4 of Fig. 5.15) while co-teaching [58] has totally failed and the heatmap is incorrect (columns 5 and 6 of Fig. 5.15). This pair belongs to the scatter plot "Patient 66009" in Fig. 5.14.

- The second row of Fig. 5.15: both methods have reasonable heatmaps, but that of the strongly-supervised classifier is smoother. This pair belongs to "Patient 78009" in the 7-th row of Fig. 5.14.

- The third row of Fig. 5.15: The heatmap of the strongly-supervised classifier is reasonable (columns 3 and 4) but that of co-teaching only picks a small spot (columns 5 and 6). This pair belongs to "Patient 501009" in the last row of Fig 5.14.

- The fourth row of Fig. 5.15: similar to the second row. This pair belongs to "Patient 19009" in the third row of Fig.5.14 where the violin plot of the strongly-supervised classifier is slightly better than that of co-teaching.

- The fifth row of Fig. 5.15: Interestingly the heatmap of co-teaching is better than that of the strongly-supervised classifier. This pair belongs to "Patient 150009" in the second last row of Fig. 5.14 where the violin plot of co-teaching is also slightly better.

- The sixth row of Fig. 5.15: both methods have failed. This failure is also visible in the violin plots "Patient 233009" in the last row of Fig. 5.14.

5.6 Conclusion

In this chapter we firstly presented our inhouse dataset called IHC4BC for predicting the status of four breast cancer biomarkers from H&E histopathology images. We showed that with a large dataset of 3K by 3K patches it is possible to achieve prediction performances around and above 90 in terms of AUC. Moreover, we showed that doing so is not possible with WSI-level labels and WSI-level prediction evaluation. Finally, we manually acquired near 900K manual spots on HER2 positive regions and demonstrated that a state-of-the-art localization method achieves competitive (and intriguingly sometimes superior) performance compared to a strongly-supervised classifier. Notably, automatic localization was ineffective (resp. effective) when using WSI-level (resp. 3K by 3K level) labels.

Chapter 6

GPEX, a Framework for Interpreting Artificial Neural Networks

6.1 Synopsis

In this section we propose a method that interprets an ANN (artificial neural network) by distilling knowledge from the ANN to a GP (Gaussian process) thereby finding globally faithful GPs which provide reliable interpretations.

6.2 Overview

ANNs (Artificial neural networks) are widely adopted in machine learning. Despite their benefits, ANNs are known to be black-box to humans, meaning that their inner mechanism for making predictions is not necessarily interpretable/explainable to humans. ANN's black-box property impedes its deployment in safety-critical applications like medical imaging or autonomous driving, and makes them hard to troubleshoot for machine learning researchers.

Attribution-based explanation methods like LIME[115], SHAP[90] and most gradient-based explanation methods like DeepLIFT [35] presume a linear surrogate model. Given a test instance $\boldsymbol{x_{test}}$, this simpler surrogate model is encouraged to have the same output "locally" around $\boldsymbol{x_{test}}$. Because of this "local assumptions", explanations from these methods might be unreliable, and can be easily manipulated by an adversary model [56, 112]. Moreover, these models may produce discordant explanations for a fixed model and test instance [78].

Considering GPs (Gaussian processes) [107] as the explainer model is beneficial, because: 1. Gaussian processes are highly interpretable. 2. Researchers have long known that GP's posterior

has the potential to match an ANN's output "globally". More precisely, given an ANN and some requirements on it [98, 50], there might exist a GP whose posterior matches the ANN's output all over the input-space $\mathbb{X}$ (as opposed to the local explanation models for which the match happens only locally around a test instance $\boldsymbol{x_{test}} \in \mathbb{X}$). Not many explainer models can globally match the ANN's output. Among gradient-based methods, with the best of our knowledge only Integrated Gradients [128] has a weak sense of ANN's global behaviour over the input space. Having some conditions on an ANN, representer point selection [136] finds a "globally faithfull" explainer model that, similar to GPs, works with a kernel function. As we will elaborate upon in Sec. 6.4.4, the GP's kernel that we find via GPEX is superior due to a technical point in the formulation of representer point selection [136]. In sum, using GPs to explain ANNs is quite promising and has advantages over other approaches to explain ANNs.

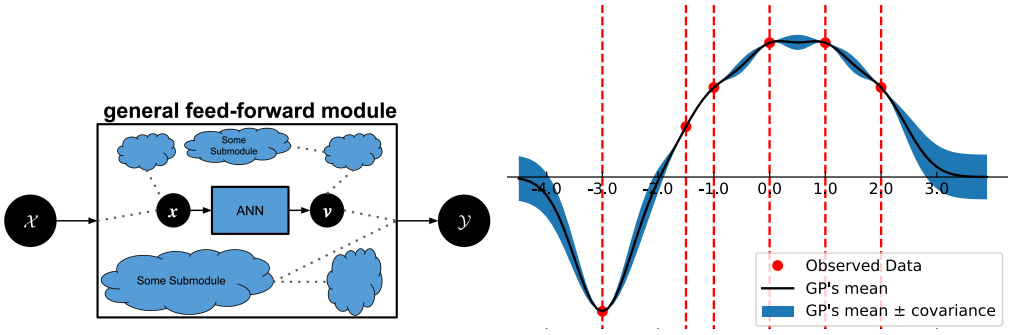
6.3 Proposed Method

6.3.1 Notation

In this section the function $g(.)$ always denotes an ANN. The kernel of a Gaussian process is denoted by the double-input function $\mathcal{K}(.,.)$. We assume that the kernel similarity between two instances $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ is equal to $f(\boldsymbol{x_i})^T f(\boldsymbol{x_j})$, where $f(.)$ maps the input-space to the kernel-space. In this chapter $\boldsymbol{u}$ (resp. v) denotes a vector in the kernel-space (resp. the posterior mean) of a GP. In some sense $\boldsymbol{u}$ and v denote the input and the output of a GP, respectively. We have that: $\mathcal{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) = f(\boldsymbol{x_i})^T f(\boldsymbol{x_j}) = \boldsymbol{u_i}^T \boldsymbol{u_j}$. The number of GPs is equal to the number of the outputs from the ANN. In other words, we consider one separate GP per scalar output-head from the ANN. We use index ℓ to specify the ℓ-th GP as follows: $\mathcal{K}_\ell(\boldsymbol{x_i}, \boldsymbol{x_j}) = f_\ell(\boldsymbol{x_i})^T f_\ell(\boldsymbol{x_j}) = \boldsymbol{u_i}^{(\ell)T} \boldsymbol{u_j}^{(\ell)}$. We parameterize the ℓ-th GP by a set of M inducing points $\{(\tilde{\boldsymbol{u}}_m^{(\ell)}, \tilde{v}_m^{(\ell)})\}_{m=1}^M$. The tilde in $(\tilde{\boldsymbol{u}}_m^{(\ell)}, \tilde{v}_m^{(\ell)})$ indicates that $\tilde{\boldsymbol{u}}$ is one of the M inducing points in the kernel space. However, $\boldsymbol{u}$ (without tilde) can be an arbitrary point in the continuous kernel-space. Notably, the inducing points $\{\tilde{\boldsymbol{x}}_m^{(\ell)}\}_{m=1}^M$ are a set of fixed instances on which data augmentation is not allowed. But $\boldsymbol{x}$ (without tilde) denotes an arbitrary vector in the feature-space that can go through data augmentation.

6.3.2 The Proposed Framework

To make our framework as general as possible, we consider a general feed-forward pipeline that contains an ANN as a submodule. In Fig. 6.1a the bigger square illustrates the general module. The input-output of the general pipeline are denoted in Fig. 6.1a by $\mathcal{X}$ and $\mathcal{Y}$. The general pipeline has at least one ANN submodule to be explained by our proposed GPEX. Fig. 6.1a illustrates this ANN by the small blue rectangle within the general pipeline. The input-output of the ANN



a b

Figure 6.1: a) A general feed-forward pipeline, with an ANN sub-module to be explained by GPEX. b) Typical behaviour of Guassian process posterior given a set of observed values.

are denoted in Fig. 6.1a by $\boldsymbol{x}$ and $\boldsymbol{v}$. Note that $\mathcal{X}$ and $\mathcal{Y}$ can be anything, including without any limitation, a set of vectors, labels, and meta-information. However, input-output of the ANN (i.e. $\boldsymbol{x}$ and $\boldsymbol{v}$) are required to be in tensor format. The exact requirements are provided in the online documentation for GPEX [14]. Moreover, the general module can have other arbitrary submodules, which are depicted by the blue clouds. The relations between the submodules, as illustrated by the dotted-lines in Fig. 6.1a, can also be quite general. Our probabilistic formulation only needs access to the conditional distributions $p(\boldsymbol{x}|\mathcal{X})$ and $p(\mathcal{Y}|\boldsymbol{x},\mathcal{X})$. Similarly, the proposed GPEX is completely agnostic about the general pipeline and it only requires the ANN's input-output to be in the tensor format. Given a PyTorch module, the proposed GPEX tool [4] automatically grabs the distributions $p(\boldsymbol{x}|\mathcal{X})$ and $p(\mathcal{Y}|\boldsymbol{x},\mathcal{X})$ from the main module it is given.

The inducing points $\{\tilde{\boldsymbol{u}}_m^{(\ell)}, \tilde{v}_m^{\ell}\}_{m=1}^M$ parameterize the ℓ-th GP. Note that $\tilde{\boldsymbol{u}}_m^{(\ell)} = f_\ell(\tilde{\boldsymbol{x}}_m)$. A feature point like $\boldsymbol{x}$ is first mapped to the kernel-space as $\boldsymbol{u}^{(\ell)} = f_\ell(\boldsymbol{x})$. Note that the kernel functions $\{f_\ell(.)\}_{\ell=1}^L$ are implemented as separate neural networks, or for the sake of efficiency as a single neural network backbone with L different heads. Afterwards, the GP's posterior on $\boldsymbol{x}$ depends on the kernel similarities between $\boldsymbol{u}^{(\ell)}$ and the inducing points $\{\tilde{\boldsymbol{u}}_m^{(\ell)}\}_{m=1}^M$. More precisely, the posterior of the ℓ-th GP on $\boldsymbol{x}$ is a random variable $v^{(\ell)}$ whose distribution is as follows [107]:

$$p(v^{(\ell)}|\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}) = \mathcal{N}\Big(v^{(\ell)}\ ;\ \mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}),\ cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)})\Big), \tag{6.1}$$

where $\mu_v(.,.,.)$ and $cov_v(.,.,.)$ are the mean and covariance of a GP's posterior computed as:

$$\mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}) = \mathcal{K}(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)})\big[\mathcal{K}(\tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}) + \sigma_{gp}^2 \mathbf{I}_{M\times M}\big]^{-1} \tilde{v}_{1:M}^{(\ell)} \tag{6.2}$$

and

$$cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M}) = \mathcal{K}(\boldsymbol{u}^{(\ell)}, \boldsymbol{u}^{(\ell)}) - \mathcal{K}(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}) \times \big[\mathcal{K}(\tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}) + \sigma^2_{gp}\mathbf{I}_{M\times M}\big]^{-1}\mathcal{K}(\tilde{\mathbf{u}}^{(\ell)}_{1:M}, \boldsymbol{u}^{(\ell)}). \tag{6.3}$$

Note that Eqs. 6.2 and 6.3 are the closed form GP posterior formulation [107], hence here we do not provide any derivation or intuitive explanation for them. Since the variables $\{v^{(\ell)}_m\}^M_{m=1}$ and v are latent or hidden, we train the model parameters by optimizing a variational lower-bound. We consider the following variational distributions:

$$q_1(v^{(\ell)}\,|\boldsymbol{x}) = \mathcal{N}\big(v^{(\ell)}\ ;\ g_\ell(\boldsymbol{x})\ ,\ \sigma^2_g\big), \qquad q_2\big(\tilde{v}^{(\ell)}_m\big) = \mathcal{N}\big(\tilde{v}^{(\ell)}_m\ ;\varphi^{(\ell)}_m, \sigma^2_\varphi\big). \tag{6.4}$$

In Eq. 6.4, the function $g_\ell(.)$ is the ℓ-th output from the ANN. Note that as the set of hidden variables $\{\tilde{v}^{(\ell)}_m\}^M_{m=1}$ is finite, we have parameterized their variational distribution by a finite set of numbers $\{\varphi^{(\ell)}_m\}^M_{m=1}$. However, as the variables $\boldsymbol{x}$ can vary arbitrarily in the feature space, the variable $\boldsymbol{u}^{(\ell)}$ varies arbitrarily in the kernel space. Therefore, the set of values $v^{(\ell)}$ may be infinite. Accordingly, the variational distribution for $v^{(\ell)}$ is conditioned on $\boldsymbol{x}$ and is parameterized by the ANN $g(.)$.

6.3.3 The Derived Evidence Lower-Bound (ELBO)

The details of the derivation of the lower-bound is moved to Appendix B. In this section we only introduce the derived ELBO and discuss how it relates the GP, the ANN and the training cost of the main module in an intuitive way. The ELBO terms containing the GP parameters (i.e. the parameters of the kernel function $f(.)$) is denoted by $\mathcal{L}_{gp}$. According to Eq. B.9 in Appendix B, $\mathcal{L}_{gp}$ is as follows:

$$\mathcal{L}_{gp} = -\frac{1}{2}\,\mathbb{E}_{\sim q}\big[\sum_{\ell=1}^{L}\frac{(\mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M}) - g_\ell(\boldsymbol{x}))^2 + \sigma^2_g}{cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M})}\big]$$
$$-\frac{1}{2}\,\mathbb{E}_{\sim q}\big[\sum_{\ell=1}^{L}\log(\frac{cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M})}{\sigma^2_g})\big] + \big(\text{const.}\big), \tag{6.5}$$

where $q(.)$ is the variational distribution that factorizes to the $q_1(.)$ and $q_2(.)$ distributions defined in Eq. 6.4. In the first term of Eq. 6.5, the numerator encourages the GP and the ANN to have the same output. More precisely, for a feature point $\boldsymbol{x}$ we can compute the corresponding point in the kernel space as $\boldsymbol{u}^{(\ell)} = f_\ell(\boldsymbol{x})$ and then compute the GP's posterior mean based on kernel similarities between $\boldsymbol{u}$ and the inducing points to get the GP's mean μ_v. In Eq. 6.5 the GP's mean μ_v is encouraged to match the ANN's output $g_\ell(\boldsymbol{x})$. In Eq. 6.5, because of the denominator of the first

term, the ANN-GP similarity is not encouraged uniformly over the feature-space. Wherever the GP's uncertainty is low, the term $cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M})$ in the denominator becomes small. Therefore, the GP's mean is highly encouraged to match the ANN's output. On the other hand, in regions where the GP's uncertainty is high, the GP-ANN analogy is less encouraged. This formulation is quite intuitive according to the behaviour of Gaussian processes. Fig. 6.1b illustrates the posterior of a GP with radial-basis kernel for a given set of observations. In regions like $[3, \infty)$ and $(-\infty, -4]$ there are no nearby observed data. Therefore, in these regions the GP is highly uncertain and the blue uncertainty margin is thick in such regions. Intuitively, our derived ELBO in Eq. 6.5 encourages the GP-ANN analogy only when GP's uncertainty is low and gives less importance to regions similar to $[3, \infty)$ and $(-\infty, -4]$ in Fig. 6.1b. Note that this formulation makes no difference for the ANN as ANNs are known to be global approximators. However, this formulation makes a difference when training the GP, because the GP is not required to match the ANN in regions where there are no similar training instances. The ELBO terms containing the ANN parameters is denoted by $\mathcal{L}_{ann}$. According to Eq. B.15 in Appendix B, $\mathcal{L}_{ann}$ is as follows:

$$\mathcal{L}_{ann} = -\frac{1}{2}\,\mathbb{E}_{\sim q}\big[\sum_{\ell=1}^{L} \frac{(\mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M}) - g_\ell(\boldsymbol{x}))^2}{cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}^{(\ell)}_{1:M}, \tilde{v}^{(\ell)}_{1:M})}\big] + \mathbb{E}_{\sim q}\big[\log p(\mathcal{Y}|\boldsymbol{y}, \mathcal{X})\big]. \tag{6.6}$$

In the above objective the first term encourages the ANN to have the same output as the GP. Similar to the objective of Eq. 6.5, the denominator of the first term gives more weight to ANN-GP analogy when GP's uncertainty is low. In the right-hand-side of Eq. 6.6, the second term is the likelihood of the pipeline's output(s), i.e. $\mathcal{Y}$ in Fig. 6.1a. This term can be, e.g., the cross-entropy loss when $\mathcal{Y}$ contains class scores in a classification problem, or the mean-squared error when $\mathcal{Y}$ is the predicted value for a regression problem, or a combination of those costs in a multi-task setting.

6.3.4 Algorithm

We consider a separate Gaussian process for each output head of an ANN. In other words, given an ANN we have as many GPs as the number of the ANN's output heads. To explain an ANN, we find the explainer GPs by optimizing the objective in Eq. 6.5 w.r.t. to the kernel mappings $\{f_\ell(.)\}_{\ell=1}^{L}$. To do so, we need to have μ_v which in turn means we need to have all kernel-space representations $\{\tilde{\boldsymbol{u}}^{(\ell)}_m\}_{m=1}^{M}$. However, it is computationally prohibitive to feed thousands of inducing instances (*i.e.* images) to the kernel mappings as $\tilde{\boldsymbol{u}}^{(\ell)}_m = f_\ell(\tilde{\boldsymbol{x}}_{\boldsymbol{m}})$ for $m \in \{1, 2, ..., M\}$ in each gradient descent iteration. On the other hand, as the kernel-space mappings $\{f_\ell(.)\}_{\ell=1}^{L}$ keep changing during training, we need to somehow track how the inducing points $\{\tilde{\boldsymbol{u}}^{(\ell)}_m\}_{m=1}^{M}$ change during training. To this end, we put the kernel-space representations of the inducing points in matrices denoted by $\mathbf{U}$. During training, these matrices are repeatedly updated by feeding mini-batches of inducing instances to the kernel-mappings.

Algorithm 1 Method Forward_GP

Input: Input instance $\boldsymbol{x}$, inducing instance $\tilde{\boldsymbol{x}}$, list of matrices $\mathbf{U}$, list of vectors $\mathbf{V}$, current kernel-space mappings $[f_1(.), ..., f_L(.)]$.

Output: List of GP posterior means μ, and covariances cov.

1: *Initialisation* : $\mu = list(L)$, $cov = list(L)$. ▷ Create two empty lists of length L.
2: **for** $\ell = 1$ to L **do**
3: $u = f_\ell(\boldsymbol{x})$ ▷ map $\boldsymbol{x}$ to the kernel space of the ℓ-th GP.
4: $\mathbf{U}_\ell \leftarrow \mathbf{U}[L]$ ▷ get the inducing points of the ℓ-th GP.
5: $\mathbf{V}_\ell \leftarrow \mathbf{V}[L]$ ▷ observed values at the inducing points.
6: **if** training **then**
7: $\mathbf{U}_\ell[\tilde{\boldsymbol{x}}.index] \leftarrow f_\ell(\tilde{\boldsymbol{x}})$ ▷ to pass gradient w.r.t. $f_\ell(.)$
8: **end if**
9: $\mu[\ell] \leftarrow \boldsymbol{u}^T\mathbf{U}_\ell^T\big(\mathbf{U}_\ell\mathbf{U}_\ell^T + \sigma_{gp}^2\mathbf{I}\big)^{-1}\mathbf{V}_\ell$. ▷ GP posterior mean formula in Eq.6.2
10: $cov[\ell] \leftarrow \boldsymbol{u}^T\boldsymbol{u} - \boldsymbol{u}^T\mathbf{U}_\ell^T\big(\mathbf{U}_\ell\mathbf{U}_\ell^T + \sigma_{gp}^2\mathbf{I}\big)^{-1}\mathbf{U}_\ell\boldsymbol{u}$. ▷ GP posterior covariance formula in Eq.6.3
11: **end for**
12: return μ and cov

Algorithm 2 Method Init_GPparams

Input: Dataset of inducing instances $[\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_M]$, current kernel-space mappings $[f_1(.), ..., f_L(.)]$, the neural network $g(.)$.

Output: List of matrices $\mathbf{U}$, list of vectors $\mathbf{V}$.

1: *Initialisation* : $\mathbf{U} = list(L)$, $\mathbf{V} = list(L)$. ▷ Create two empty lists of length L.
2: **for** $\ell = 1$ to L **do**
3: $\mathbf{V}[\ell] \leftarrow [g(\tilde{\boldsymbol{x}}_1)[\ell], ..., g(\tilde{\boldsymbol{x}}_M)[\ell])]$.
4: **end for**
5: **for** $\ell = 1$ to L **do**
6: $\mathbf{U}[\ell] \leftarrow [f_\ell(\tilde{\boldsymbol{x}}_1), ..., f_\ell(\tilde{\boldsymbol{x}}_M)]$.
7: **end for**
8: return $\mathbf{U}$ and $\mathbf{V}$

Alg. 4 optimizes the objective of Eq. 6.5 w.r.t. the kernel mappings $\{f_\ell(.)\}_{\ell=1}^L$. First, a single training instance $\boldsymbol{x}$ and a single inducing point $\tilde{\boldsymbol{x}}$ are selected (line 3-4). Afterwards, the procedure of Alg. 3 is called to update the kernel mappings (line 5 of Alg. 4). To update the kernel mappings, the GP posterior is computed via the matrices $\mathbf{U}$ (line 2 of Alg.3). The "forward_GP" procedure (called in line 2 of Alg. 3) is provided in Alg. 1, and uses the matrices $\mathbf{U}$ to compute GP's posterior. Only the rows of $\mathbf{U}$ that correspond to the selected inducing point $\tilde{\boldsymbol{x}}$ are computed using the kernel-mappings, so that the gradient w.r.t. the kernel-mappings can be computed in the backward pass (lines 6-7 of Alg. 1). In other words, to avoid the infeasible GPU memory requirement when creating the computation graph, only $\boldsymbol{u} = f(\boldsymbol{x})$ and $\tilde{\boldsymbol{u}} = f(\tilde{\boldsymbol{x}})$ are computed using the actual kernel mappings, and for the rest of inducing points the current representations in the $\boldsymbol{U}$ matrices are used. Finally, the matrices $\mathbf{U}$ are updated (lines 7-9 of Alg. 4). Of course instead of a single training/inducing instance, we used a mini-batch of multiple training/inducing instances.

Algorithm 3 Method Update_KernMappings

Input:

Input instance $\boldsymbol{x}$, inducing instance $\tilde{\boldsymbol{x}}$, list of matrices $\mathbf{U}$, list of vectors $\mathbf{V}$, current kernel-space mappings $[f_1(.), ..., f_L(.)]$.

Output:

New kernel-space mappings $[\hat{f}_1(.), ..., \hat{f}_L(.)]$, after one step of gradient descent.

Initialisation : $loss \leftarrow 0$.
μ, $cov \leftarrow$ forward_GP($\mathbf{x}$, $\tilde{\boldsymbol{x}}$, $\mathbf{U}$, $\mathbf{V}$) ▷ feed $\mathbf{x}$ to GPs, "forward_GP" is defined in Alg. 1.
$\mu_{ann} \leftarrow g(\mathbf{x})$ ▷ feed $\mathbf{x}$ to ANN.
for $\ell = 1$ to L **do**
    $loss \leftarrow loss + \frac{(\mu[\ell]-\mu_{ann}[\ell])^2+\sigma_g^2}{cov[\ell]} + \log(cov[\ell])$. ▷ Eq. 6.6.
end for
$\boldsymbol{\delta} \leftarrow \frac{\partial\ loss}{\partial\ params\left([f_1(.),...,f_L(.)]\right)}$. ▷ the gradient of loss.
$params\left([\hat{f}_1, ..., \hat{f}_L]\right) \leftarrow params\left([f_1, ..., f_L]\right) - lr \times \boldsymbol{\delta}$ ▷ update the parameters.
$lr \leftarrow$ updated learning rate
return $[\hat{f}_1(.), ..., \hat{f}_L(.)]$

6.3.5 Making the Algorithm Scalable

One major difficulty of training GPs is the matrix inversion of Eqs. 6.2 and 6.3, which has $\mathcal{O}(M^3)$ complexity using standard matrix inversion methods. To address this issue, we adopted computational techniques recently used for fast spectral clustering [61] as follows. Let $\boldsymbol{A}$ be an arbitrary $M \times D$ matrix where $M >> D$. Moreover, let $\boldsymbol{b}$ be a M-dimensional vector and let σ be a scalar. The computational techniques [61] allow us to efficiently compute:

$$(\mathbf{A}\mathbf{A}^T + \sigma^2 \boldsymbol{I}_{M\times M})^{-1}\, \boldsymbol{b}.$$

The idea is that $\mathbf{A}\mathbf{A}^T$ and therefore its inverse are of rank D. Therefore, $(\mathbf{A}\mathbf{A}^T)^{-1}$ has D non-zero eigenvalues like $\{\lambda_1, ..., \lambda_D\}$ and the rest of its eigenvalues are zero. Let the corresponding eigenvectors be $\{\boldsymbol{e_1}, ..., \boldsymbol{e_D}\}$. To compute $(\mathbf{A}\mathbf{A}^T)^{-1}\boldsymbol{b}$ we can simply project $\boldsymbol{b}$ to the D-dimensional space of the eigenvectors. By doing so, we avoid the $\mathcal{O}(M^3)$ computational complexity. Let $\{\lambda_1, ..., \lambda_D\}$ be the non-zero eigenvalues of $\mathbf{A}\mathbf{A}^T$ and let $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_D\}$ be the corresponding eigenvectors. From linear algebra, it follows that for $\mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I}_{M\times M}$ the eigenvalues and the eigenvectors are $\{\lambda_1 + \sigma^2, ..., \lambda_D + \sigma^2, \sigma^2, ..., \sigma^2\}$ and $\{\boldsymbol{e_1}, ..., \boldsymbol{e_D}\}$, respectively. Note that $M - D$ eigenvalues are added all of which are equal to σ^2. Similarly, from linear algebra it follows that for the inverse of $\mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I}_{M\times M}$ the eigenvalues and eigenvectors are $\{\frac{1}{\lambda_1+\sigma^2}, ..., \frac{1}{\lambda_D+\sigma^2}, \frac{1}{\sigma^2}, ..., \frac{1}{\sigma^2}\}$ and $\{\boldsymbol{e_1}, ..., \boldsymbol{e_D}, \boldsymbol{e_{D+1}}, ..., \boldsymbol{e_M}\}$ respectively. Note that although there are M eigenvectors, only the first D eigenvectors appear in our computations. More precisely, let $\mathbf{E} \in \mathbb{R}^{M\times D}$ be a matrix whose columns are $\{\boldsymbol{e_1}, ..., \boldsymbol{e_D}\}$. Let $\boldsymbol{\Lambda}$ be a diagonal matrix whose diagonal is formed by $\{\frac{1}{\lambda_1+\sigma^2}, ..., \frac{1}{\lambda_D+\sigma^2}\}$. In the space of the D eigenvectors the linear transformation on any vector like $\boldsymbol{b}$ is equal to $\mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T\boldsymbol{b}$,

Algorithm 4 Method Explain_ANN

Input: Training dataset *ds_train*, the inducing dataset *ds_inducing*, the neural network $g(.)$.
Output: The obtained GPs, *i.e.*, the kernel-space mappings $[f_1(.), ..., f_L(.)]$, list of matrices $\mathbf{U}$, and list of vectors $\mathbf{V}$.

Initialisation : $\mathbf{U}$, $\mathbf{V}$ $\leftarrow$ Init_GPparams(ds_inducing) ▷ the procedure of Alg. 2
for *iter* $= 1$ to *max_iter* **do**
  $\boldsymbol{x} \leftarrow randselect(ds\_train)$.
  $\tilde{\boldsymbol{x}} \leftarrow randselect(ds\_inducing)$
  $[f_1(.), ..., f_L(.)]$ $\leftarrow$ Update_KernMapings$(\boldsymbol{x}, \tilde{\boldsymbol{x}}, \mathbf{U}, \mathbf{V})$. ▷ the procedure of Alg. 3
  $\tilde{\boldsymbol{x}} \leftarrow randselect(ds\_inducing)$.
  for $\ell = 1$ to L **do**
    $U[\ell][\tilde{\boldsymbol{x}}.index] \leftarrow f_\ell(\tilde{\boldsymbol{x}})$ ▷ update kernel-space representations.
  end for
end for
return $[f_1(.), ..., f_L(.)]$, $\mathbf{U}$, $\mathbf{V}$

Algorithm 5 Method Efficiently_Compute_AATinvb

Input: Matrix $\mathbf{A}$ of size $M \times D$, vector $\boldsymbol{b}$ of size $M \times 1$, and positive scalar σ.
Output: The vector **output** $= (\mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I})^{-1}\boldsymbol{b}$.

$\tilde{\mathbf{E}}, \tilde{\boldsymbol{\lambda}} \leftarrow eigendecomp\big(\mathbf{A}^T\mathbf{A} + \sigma^2\mathbf{I}\big)$.
$[\tilde{\boldsymbol{e}}_1, ..., \tilde{\boldsymbol{e}}_D] \leftarrow \tilde{\mathbf{E}}$
$[\tilde{\lambda}_1, ..., \tilde{\lambda}_D] \leftarrow \tilde{\boldsymbol{\lambda}}$
$[\boldsymbol{e}_1, ..., \boldsymbol{e}_D] \leftarrow [\mathbf{A}\tilde{\boldsymbol{e}}_1, ..., \mathbf{A}\tilde{\boldsymbol{e}}_D]$
$[\lambda_1, ..., \lambda_D] \leftarrow [\tilde{\lambda}_1, ..., \tilde{\lambda}_D]$
$\mathbf{E} \leftarrow [\boldsymbol{e}_1, ..., \boldsymbol{e}_D]$
$\boldsymbol{\Lambda} \leftarrow diagonal(\frac{1}{\lambda_1+\sigma^2}, ..., \frac{1}{\lambda_D+\sigma^2})$
output $\leftarrow \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T\boldsymbol{b} + \frac{1}{\sigma^2}(\boldsymbol{b} - \mathbf{E}\mathbf{E}^T\boldsymbol{b})$ ▷ according to Eq. 6.7
return **output**

meaning that multiplication by $\mathbf{E}^T$ transforms $\boldsymbol{b}$ to the space of the D eigenvectors, multiplication by $\boldsymbol{\Lambda}$ performs the transformation in that space, and multiplication by $\mathbf{E}$ transforms the result back to the original space. The $(M - D)$ eigenvalues that correspond to the rest of the eigenvectors are all the same and are equal to $\frac{1}{\sigma^2}$. Therefore, there is no need to project $\boldsymbol{b}$ to the space of the $(M - D)$ eigenvectors because the linear transformation in that space is simply a scaling by $\frac{1}{\sigma^2}$. All in all, we have that

$$\big(\mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I}_{M\times M}\big)^{-1}\boldsymbol{b} \;=\; \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T\boldsymbol{b} + \frac{1}{\sigma^2}(\boldsymbol{b} - \mathbf{E}\mathbf{E}^T\boldsymbol{b}). \tag{6.7}$$

Complexity of computing the right-hand-side of Eq. 6.7 is way lower than the $\mathcal{O}(M^3)$ requirement of the standard matrix inversion. We borrowed more computational ideas from the work on fast spectral clustering [61]. To compute the first D eigenvlaues and eigenvectors of $\mathbf{A}\mathbf{A}^T$, we worked

with the D-by-D matrix $\mathbf{A}^T\mathbf{A}$ rather than the M-by-M matrix $\mathbf{A}\mathbf{A}^T$ (recall that $D << M$), because given the eigenvalues and eigenvectors of $\mathbf{A}^T\mathbf{A}$, those of $\mathbf{A}\mathbf{A}^T$ are easily computable [61]. The procedure is explained in Alg. 5. In Alg. 5, lines 1-3 compute the eigenvalues/vectors of the matrix $\mathbf{A}^T\mathbf{A}$. Afterwards, lines 4 and 5 compute the first D eigenvalues/vectors of $\mathbf{A}\mathbf{A}^T$ using those of $\mathbf{A}^T\mathbf{A}$. Finally, line 8 computes $(\mathbf{A}\mathbf{A}^T + \sigma^2\mathbf{I})^{-1}\boldsymbol{b}$ according to the right-hand-side of Eq. 6.7. To make the computations faster, we made use of the following equation $\mathbf{A}\mathbf{A}^T = \sum_m \mathbf{A}[m,:]\mathbf{A}[m,:]^T$, where $\mathbf{A}[m,:]$ is the m-th row of the matrix $\mathbf{A}$. Thanks to this equation, we compute $\mathbf{A}\mathbf{A}^T$ only once at the beginning of the training. Afterwards, as each mini-batch alters only some rows of $\mathbf{A}$, we update the previously computed $\mathbf{A}\mathbf{A}^T$ by considering only the effect of the modified rows.

6.3.6 An Important Note on How to Use Dataset Instances

As we will see in Sec. 6.4.6, the inducing dataset (*i.e.* "ds_inducing" in Alg. 4) should be as large as possible so the GP posteriors can be flexible enough to match the ANNs [133]. Therefore a good practice is to include all training instances (without data augmentation) in "ds_inducing". But by doing so the following issue arises. An instance from "ds_train" like $\boldsymbol{x}$ is an augmented version of an inducing instance $\tilde{\boldsymbol{x}}$. Because $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are close, their kernel-space representations $f(\boldsymbol{x})$ and $f(\tilde{\boldsymbol{x}})$ also become close regardless of parameters of $f(.)$. Consequently, regardless of $f(.)$, GP's posterior mean will be roughly equal for both $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. Indeed, in this case Alg. 4 fails to find the kernel mappings $\{f_\ell(.)\}_{\ell=1}^L$. To avoid this issue, we sample $\boldsymbol{x}$ in line 3 of Alg. 4 as follows: $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are randomly selected from "ds_train", and $\alpha \sim uniform(-1, 2)$, and $\boldsymbol{x} = \alpha\boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2$. The rest of Alg. 4 after line 3 is run as before.

6.3.7 Computing Pixel-level Contributions to Similarities

We first explain the idea of CAM [139], afterwards we modify it for the architectures of our kernel modules. Let the kernel mapping $f(.)$ be a convolutional neural network that produces a volumetric map of size $C \times H \times W$ followed by a spatial average pooling that produces the C-dimensional vector in the kernel-space. In this case, $\mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is as follows:

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= f(\boldsymbol{x}_1)^T\, f(\boldsymbol{x}_2) \\ &= \Big(\sum_{i=1}^{H}\sum_{j=1}^{W} \boldsymbol{z}_{ij}^{(1)}\Big)^T \Big(\sum_{k=1}^{H}\sum_{\ell=1}^{W} \boldsymbol{z}_{k\ell}^{(2)}\Big) \\ &= \sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{H}\sum_{\ell=1}^{W} \big(\boldsymbol{z}_{ij}^{(1)^T} \boldsymbol{z}_{k\ell}^{(2)}\big),\end{aligned} \tag{6.8}$$

where $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$ are the volumetric maps of size $C \times H \times W$ and the indices (i, j) and (k, ℓ) index the spatial locations over the volumetric maps. The last term in Eq. 6.8 shows that the total similarity $\mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is the sum of the contributions from each pair of positions (i, j) on $\boldsymbol{x}_1$ and (k, ℓ) on $\boldsymbol{x}_2$. To compute the contribution of a specific location like (i, j) on $\boldsymbol{x}_1$, we sum up the contributions of (i, j) on $\boldsymbol{x}_1$ and all possible locations $\{(k, \ell)\}_{k=1}^{H}{}_{\ell=1}^{W}$ on $\boldsymbol{x}_2$.

The kernel-mappings that we used have a slightly different architecture than a volumetric map followed by spatial average pooling. Our kernel mappings produce a volumetric map of size $C \times H \times W$ followed by a spatial average pooling that produces a C-dimensional vector. Afterwards, the resulting vector is divided by its ℓ_2-norm to produce a vector of norm 1. Consequently, this vector of norm 1 is fed to a leaky ReLU layer that produces the final kernel-space representation $f(\boldsymbol{x})$. For this architecture the pixel contributions can be computed according to an equation similar to Eq. 6.8 as follows. Our kernel mappings produce the volumetric map $\boldsymbol{z}$ of size $C \times H \times W$ followed by a spatial average pooling that produces the C-dimensional vector $\boldsymbol{a}$:

$$\boldsymbol{a} = \sum_{i=1}^{H} \sum_{j=1}^{W} \boldsymbol{z}_{ij}. \tag{6.9}$$

Afterwards, the resulting vector is divided by its ℓ_2-norm to produce the vector $\boldsymbol{b}$ of norm 1:

$$\boldsymbol{b} = [\frac{a_1}{||\boldsymbol{a}||_2}, \ \dots \ , \frac{a_C}{||\boldsymbol{a}||_2}]. \tag{6.10}$$

Consequently, this vector of norm 1 is fed to a leaky ReLU layer that produces the final kernel-space representation $f(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = leakyReLU(\boldsymbol{b}). \tag{6.11}$$

We begin with simplifying Eq. 6.11. The leaky ReLU activation function multiplies the input by a constant and this constant depends on the sign of the input. Therefore, applying the leaky ReLU activation is equivalent to multiplication by a diagonal matrix $\boldsymbol{\Lambda}$. Therefore,

$$f(\boldsymbol{x}) = \boldsymbol{\Lambda b}. \tag{6.12}$$

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be two images, and $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$ be the corresponding volumetric maps. We have that

$$\boldsymbol{a}^{(1)} = \sum_{i=1}^{H} \sum_{j=1}^{W} \boldsymbol{z}_{ij}^{(1)}, \quad \boldsymbol{a}^{(2)} = \sum_{k=1}^{H} \sum_{\ell=1}^{W} \boldsymbol{z}_{k\ell}^{(2)}. \tag{6.13}$$

And

$$\boldsymbol{b}^{(1)} = [\frac{a_1^{(1)}}{||\boldsymbol{a}^{(1)}||_2}, \ \dots \ , \frac{a_C^{(1)}}{||\boldsymbol{a}^{(1)}||_2}], \quad \boldsymbol{b}^{(2)} = [\frac{a_1^{(2)}}{||\boldsymbol{a}^{(2)}||_2}, \ \dots \ , \frac{a_C^{(2)}}{||\boldsymbol{a}^{(2)}||_2}]. \tag{6.14}$$

And

$$f(\boldsymbol{x}^{(1)}) = \boldsymbol{\Lambda}^{(1)}\boldsymbol{b}^{(1)}, \quad f(\boldsymbol{x}^{(2)}) = \boldsymbol{\Lambda}^{(2)}\boldsymbol{b}^{(2)}. \tag{6.15}$$

Now we simplify the similarity $\mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2)$:

$$\begin{aligned}
\mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \big(\boldsymbol{\Lambda}^{(1)}\boldsymbol{b}^{(1)}\big)^T\big(\boldsymbol{\Lambda}^{(2)}\boldsymbol{b}^{(2)}\big) \\
&= \big(\boldsymbol{\Lambda}^{(1)^T}\boldsymbol{\Lambda}^{(2)}\big)\big(\boldsymbol{b}^{(1)^T}\boldsymbol{b}^{(2)}\big) \quad \leftarrow \text{note that } \boldsymbol{\Lambda}^{(1)} \text{ and } \boldsymbol{\Lambda}^{(2)} \text{ are diagonal, hence this line follows} \\
&= \frac{\big(\boldsymbol{\Lambda}^{(1)^T}\boldsymbol{\Lambda}^{(2)}\big)}{||\boldsymbol{a}^{(1)}||_2 \ \ ||\boldsymbol{a}^{(2)}||_2} \ \big(\sum_{i=1}^{H}\sum_{j=1}^{W}\boldsymbol{z}_{ij}^{(1)}\big)^T\big(\sum_{k=1}^{H}\sum_{\ell=1}^{W}\boldsymbol{z}_{k\ell}^{(2)}\big) \\
&= \frac{\big(\boldsymbol{\Lambda}^{(1)^T}\boldsymbol{\Lambda}^{(2)}\big)}{||\boldsymbol{a}^{(1)}||_2 \ \ ||\boldsymbol{a}^{(2)}||_2} \ \sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{H}\sum_{\ell=1}^{W}\big(\boldsymbol{z}_{ij}^{(1)^T}\boldsymbol{z}_{k\ell}^{(2)}\big).
\end{aligned} \tag{6.16}$$

In sum, as our architecture for kernel-mappings is slightly different than the one that CAM [139] assumes (a volumetric map followed by spatial average pooling), instead of Eq. 6.8 we used Eq. 6.16 that we derived above.

6.4 Experiments

6.4.1 Measuring Faithfulness of GPs to ANNs

We firstly examined if the obtained Gaussian processes match the corresponding artificial neural networks, *i.e.*, if the knowledge distillation of Eq. 6.5 is performed successfully. As seen in Fig. 6.1a GPEX can replace any arbitrary ANN submodule by Gaussian processes. To test this ability, we applied GPEX to two types of ANN submodules: 1. a classifier, 2. the attention submodule of a classifier pipeline.

Experimental Setup

We conducted experiments on four publicly available datasets: MNIST [52], Cifar10 [83], Kather [77], and DogsWolves [131]. For MNIST [52] and Cifar10 [83] we used the standard split to training and test sets provided by the datasets. For Kather [77] and DogsWolves [131] we randomly selected

70% and 80% of instances as our training set. There are L kernel mappings that we denoted by $[f_1(.), f_2(.), ..., f_L(.)]$. One can implement this kernel mappings by, e.g., considering L independent CNNs. However, doing so dramatically increases the computation cost. Therefore, we modeled the L mappings by a common ResNet-50 [60] backbone. After the common backbone, we placed L branches. Each branch has two convolutional layers followed by global spatial average pooling that produce a vector. Each branch ends with an L2 normalizer layer (that sets the L2-norm of the vector to 1) followed by a leaky-ReLU layer. During our experiments we noticed that the L2-normalization layer and the final leaky-ReLU layer are essential. Without the L2 normalization layer, the vectors in the kernel-space can have arbitrarily-small or arbitrarily-big elements, and this makes the training unstable. We included the last leaky-ReLU layer, because according to GP posterior mean formula in Eq. 6.1 vectors in the kernel-space go through a linear transformation. Therefore, without the last leaky-ReLU layer, the pipeline would have two consecutive linear layers. Throughout our experiments, we set the output of each branch (i.e. vectors in the kernel-space of each GP) to be 20-dimensional.

We trained a separate convolutional neural network (CNN) on each dataset to perform the classification task. For MNIST [52], Cifar10 [83], and Kather [77] we used a ResNet-18 [60] backbone followed by some fully connected layers. DogsWolves [131] is a relatively small dataset, and very deep architectures like ResNet [60] quickly overfit to the training set. Therefore, we used a convolutional backbone which is suggested in the dataset website [131]. For all datasets, we set the width (i.e. the number of neurons) of the second last fully-connected layer to 1024. Because according to theoretical results on GP-ANN analogy [97, 50] the second last layer of ANN should be wide. We used an implementation of ResNet [60] which is publicly available online [2]. We trained the pipelines for 20, 200, 20, and 20 epochs on MNIST [52], Cifar10 [83], Kather [77], and DogsWolves [131], respectively. For Cifar10 [83], we used the exact optimizer suggested by [2]. For other datasets we used an Adam [80] optimizer with a learning-rate of 0.0001. The test accuracies of the models are equal to 99.56%, 95.43%, 96.80%, and 80.50% on MNIST [52], Cifar10 [83], Kather [77], and DogsWolves [131], respectively. The pipeline with an attention mechanism consists of two ResNet-18 [60] backbones: one extracts a volumetric map containing deep features, and the other produces a spatial attention mask. For each attention backbone we set the width of the second last layer to 1024, followed by a linear layer and sigmoid activation. When applying our proposed GPEX we used Adam optimizer [80]. Although the AMSGrad version of this optimizer is often recommended, for our proposed GPEX we noticed the Adam optimizer [80] without AMSGrad works the best. For classifier ANN submodules we used a learning-rate of 0.0001 while for the attention submodules we used a learning rate of 0.00001.

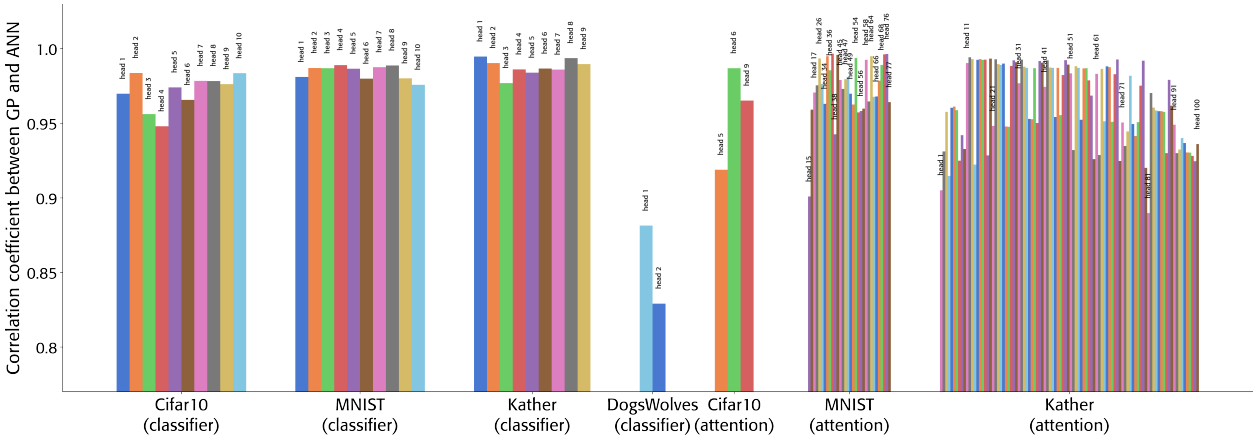


Figure 6.2: Faithfulness of GPs to ANNs measured by Pearson correlation coefficient.

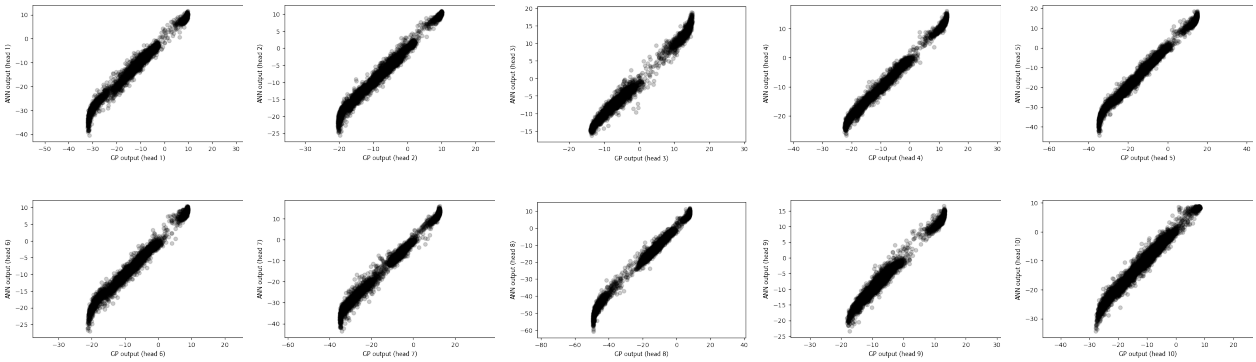


Figure 6.3: Scatters for MNIST (classifier).

Results

To measure the faithfulness of GPs to ANNs, we compute the Pearson correlation coefficient for each ANN head and the mean of the corresponding GP posterior on unseen test instances. The results are provided in Fig. 6.2. In Fig. 6.2, the first four groups of bars (i.e. the groups labeled as Cifar10 (classifier), MNIST (classifier), Kather (classifier), and DogsWolves (classifier)) correspond to applying the proposed GPEX to the four classifier ANNs trained on the four datasets. According to Fig. 6.2, our trained GPs almost perfectly match the corresponding ANNs. Only for DogsWovles [131], as illustrated by the 4-th bar group in Fig. 6.2, the correlation coefficients are lower compared to other datasets. We hypothesize that this is because the DogsWolves dataset [131] is relatively small (2K images in total) which impedes perfect knowledge distillation. In fact, GP posterior mean can be changed only by moving the inducing points in the kernel-space. Therefore, when very few inducing points are available the GP posterior mean is less flexible [133]. This is consistent with our parameter analysis in Sec. 6.4.6. In Fig. 6.2, 5-th, 6-th, and 7-th bar groups show the

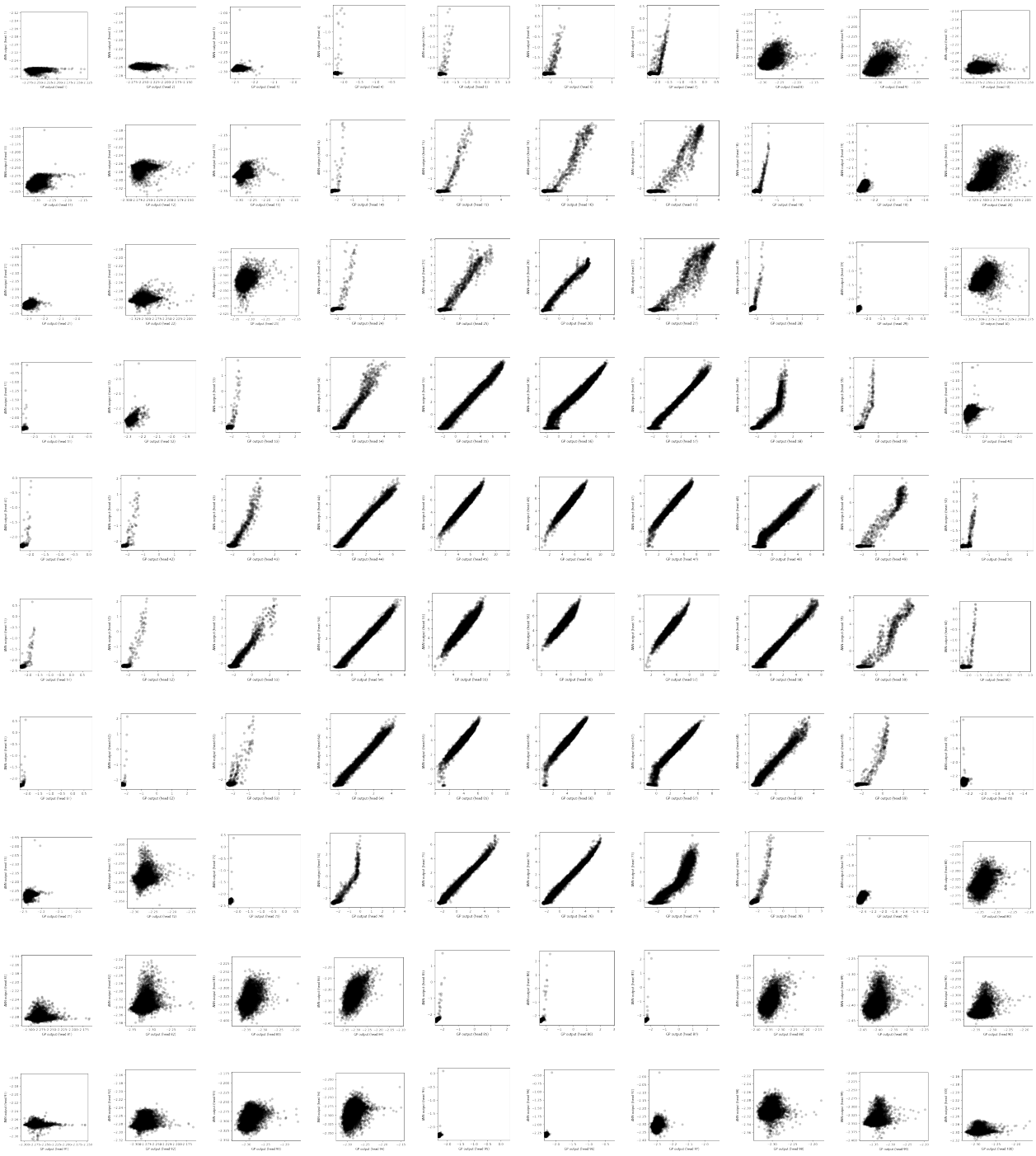


Figure 6.4: Scatters for MNIST (attention).

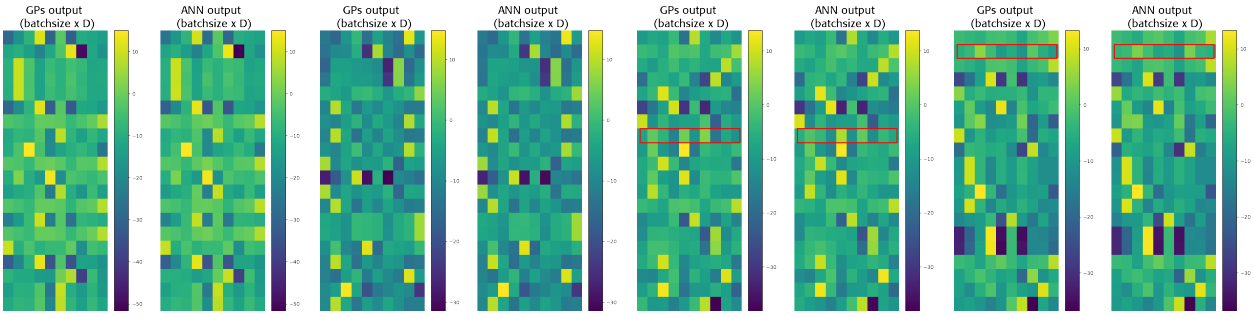


Figure 6.5: Comparing GP and ANN outputs for four batches of the MNIST [52] dataset. The red rectangles highlight the instances for which the predictions of GP and ANN (i.e. the class with maximum score) are different.

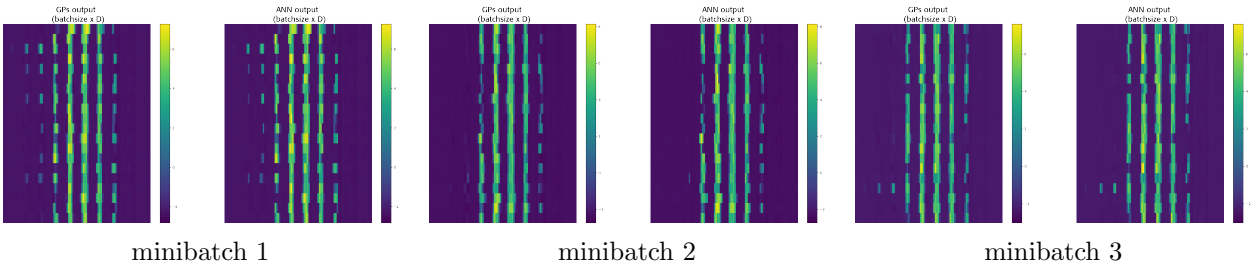


Figure 6.6: Comparing GP and ANN (attention submodule) outputs for 3 batches of the MNIST [52]dataset.

correlation coefficients between the attention backbones and the corresponding GPs on unseen test instances. According to Fig. 6.2, our proposed GPEX is able find GPs which are faithful to attention subcomponents of the classifier pipelines. Each attention submodule produces a mask of size $[h, w]$. We flattened the output mask, thereby thinking of the attention submodule as an ANN with $h \times w$ output heads.

Besides reporting Pearson correlations in Fig. 6.2, we depict scatter plots between the posterior mean of the obtained GPs and corresponding ANNs. The results are provided for MNIST dataset in Fig. 6.3 (classifier) and 6.4 (attention), and for other datasets in Appendix C, Figs. C.1, C.2, C.3, and C.4. To get more insights, we selected mini-batches of testing instances and fed each mini-batch to both ANN and corresponding GPs. The output from ANN (and similarly GPs) is a matrix of shape $batchsize \times D_v$, where D_v is the number of output heads from the ANN. Ideally we should get two identical $batchsize \times D_v$ matrices for each mini-batch, because the GPs are supposed to be faithful to ANNs. The results for MNIST dataset are provided in Fig. 6.5 (classifier) and 6.6, and for other datasets in Appendix C, Figs. C.5, C.6, C.7, C.8, and C.9.

The red rectangles in Figs. 6.5, C.5, C.6, and C.7 show the test instances for which the GP's

	Cifar10 [83]	MNIST [52]	Kather [77]	DogsWolves [131]
ANN accuracy	95.43	99.56	96.80	80.50
GPs accuracy	92.26	99.41	93.60	78.75

Table 6.1: Accuracies of ANN classifiers versus the accuracies of the explainer GPs on four datasets.

decision (i.e. the class with the highest score) does not match the ANN's decision. According to these figures the disagreement between GPs prediction and ANN prediction mostly happens when either some output activations are very close to one another or all activations are close to zero. This is consistent with the scatter plots of Figs. 6.3, C.1, and C.2 in which the scatters are slightly dispersed for intermediate values. Tab. 6.1 reports the test accuracy of the ANNs and their corresponding GPs. We see that GPs' accuracies are slightly lower than those of the corresponding ANNs. Figs. 6.5, C.5, C.6, and C.7 provide insights about how this small disagreement can be potentially solved in future research by, e.g., preventing the ANN from having near-zero activations or having output heads which are very close to one another. Note that we didn't include all attention heads in Fig. 6.2 because some pixels in attention masks (*e.g.* the blob-like scatters in Fig. 6.4) are always off for different input instances and are excluded in Fig. 6.2.

6.4.2 Explaining ANNs' Decisions

Experimental Setup

In Sec. 6.4.1 we trained four CNN classifiers on Cifar10 [83], MNIST [52], Kather [77], and DogsWolves [131] datasets, respectively. Afterwards we applied our proposed explanation method to each CNN classifier. In this section, we are going to explain the decisions made by the classifiers via the obtained GPs found by Alg. 4. We explain the decision made for a test instance like $\boldsymbol{x}_{test}$ as follows. We consider the GP and the kernel-space that correspond to the ANN's head with maximum value (i.e. the ANN's head that relates to the predicted label). Consequently, among the instances in the inducing dataset, we find the 10 closest instances to $\boldsymbol{x}_{test}$, like $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{i10}\}$. Intuitively the ANN has labeled $\boldsymbol{x}_{test}$ in that way because it has found $\boldsymbol{x}_{test}$ to be similar to $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{i10}\}$.

6.4.3 Results

For MNIST digit classification, some test instances and nearest neighbours in training set are shown in Fig. 6.7. In this figure each row corresponds to a test instance. The first column depicts the test instance itself and columns 2 to 11 depict the 10 nearest neighbours. For example, in Fig. 6.7 the image in row3-col1 depicts a test instance $\boldsymbol{x}_{test}$ and the images in row3, cols2-11 depict the nearest neighbours $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{i10}\}$. According to rows 1 and 2 of Fig. 6.7, the classifier has labeled the

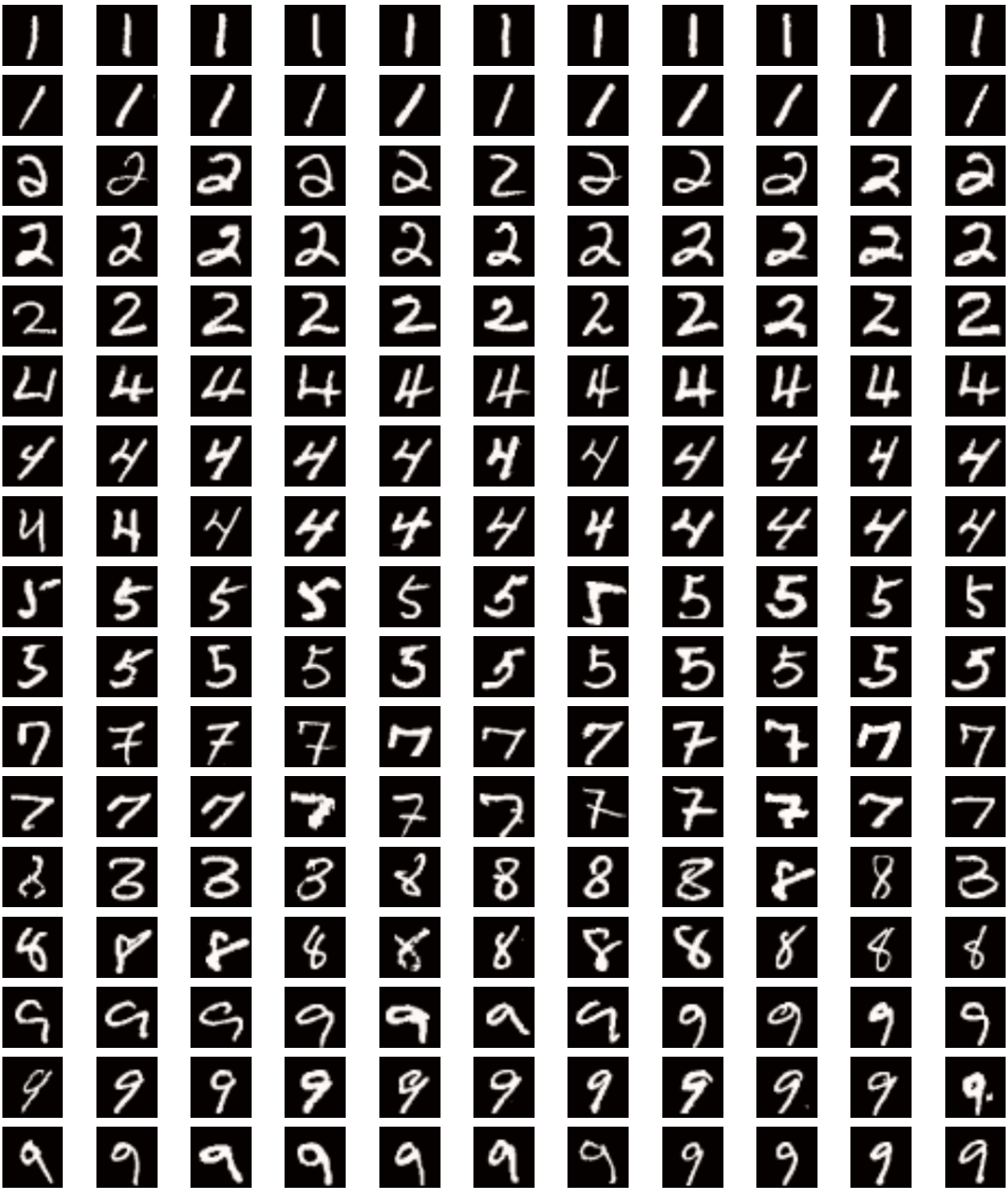


Figure 6.7: Sample explanations for MNIST dataset.

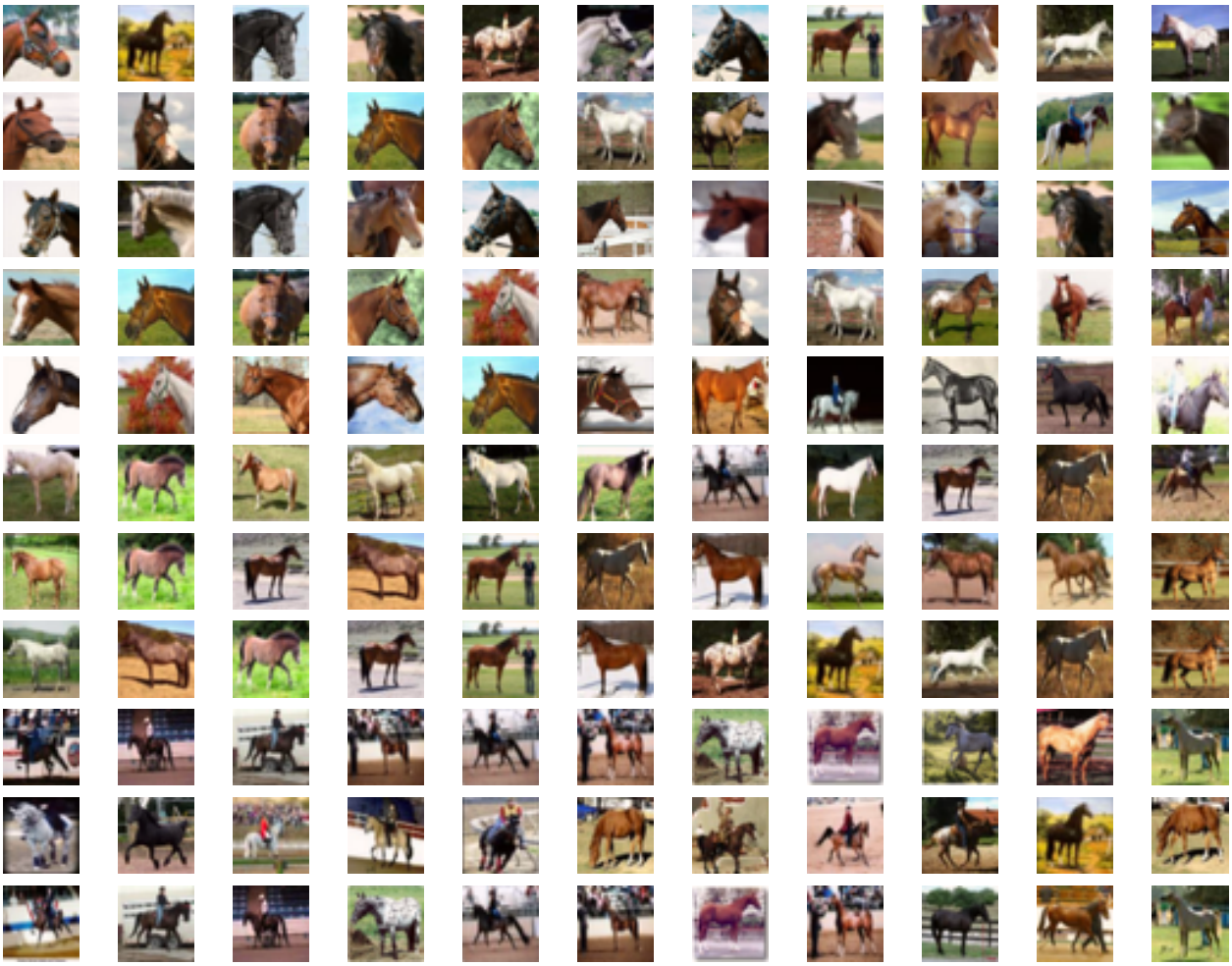


Figure 6.8: Sample explanations for Cifar10 dataset.

two images as digit 1 because it has found 1 digits with similar inclinations in the training set (in Fig. 6.7 in row 1 all digits are vertical but in row 2 all digits are inclined). We see the model has also taken the inclination into account for the test instances of rows 7, 8, 15, 16, and 17 of Fig. 6.7. In Fig. 6.7, according to rows 3, 4, and 5 the test instances are classified as digit 2 because 2 digits with similar styles are found in the training set. We see the model has also taken the style into account for the test instances of rows 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, and 17 of Fig. 6.7. For instance, the test instance in row 6 of Fig. 6.7 is a 4 digit with a short stand and the two nearest neighbours are alike. Or for the test instances in rows 13, 14, and 15 of Fig. 6.7 the test instances have incomplete circles in the same way as their nearest neighbours.

Fig. 6.8 illustrates some sample explanations for Cifar10 [83]. Like before, each row corresponds to a test instance, the first column depicts the test instance itself and columns 2 to 11 depict the

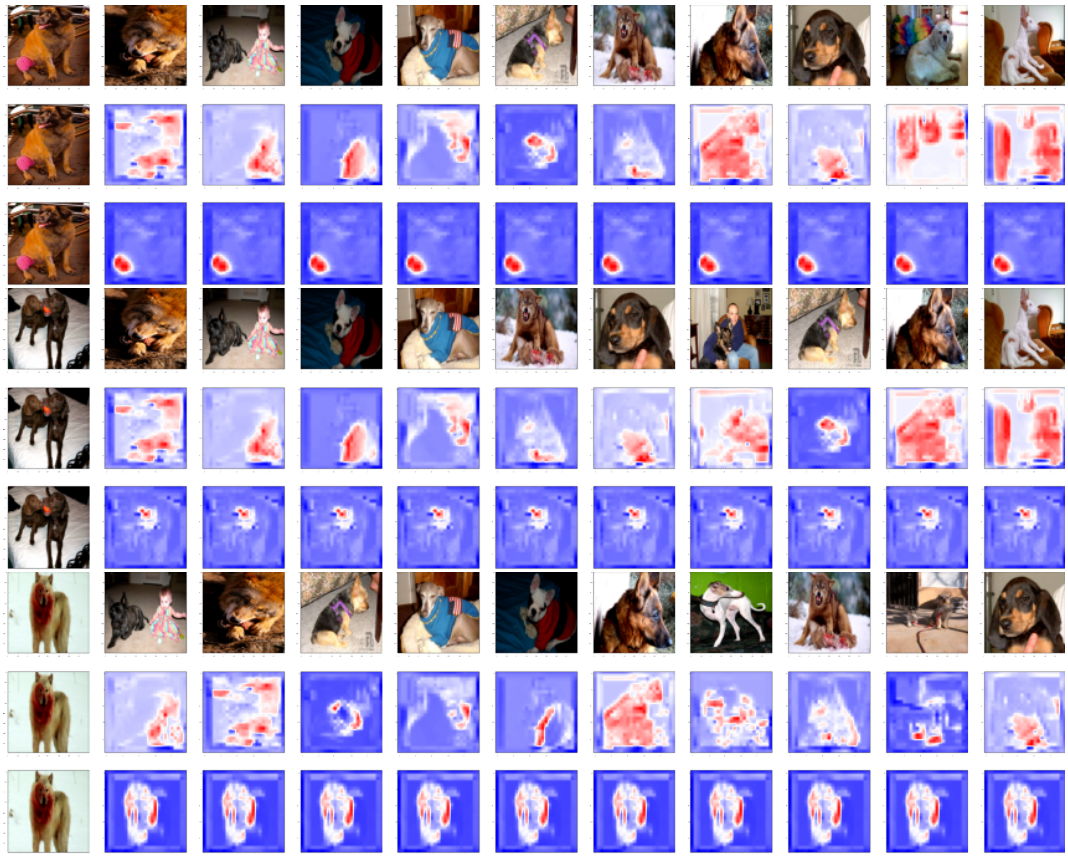


Figure 6.9: Sample explanations for DogsWolves dataset.

10 nearest neighbours. In Fig. 6.8, the test instances of rows 1, 2, 3, 4, and 5 are captured from horses' heads from closeby, and the nearest neighbours are alike. However, in rows 6, 7, 8, 9, 10, and 11 of Fig. 6.8 the test images are taken from faraway and the found similar training images are also taken from faraway. Intuitively, as the classifier is not aware of 3D geometry, it finds training images which are captured from the same distance. In rows 9, 10, and 11 of Fig. 6.8, we see that the testing images contain riders. Similarly, the nearest neighbours also tend to have riders. Intuitively, in rows 9, 10, and 11 of Fig. 6.8 the model has made use of the riders or other context information to classify the test instances as horse.

Besides finding the nearest neighbours, we provide CAM-like [139] explanations as to why $\boldsymbol{x}_{test}$ and an instance like $\boldsymbol{x}_{ij}, 1 \leq j \leq 10$ are considered similar by the model (according to the procedure of Sec. 6.3.7). Fig. 6.9 illustrates some sample explanations for DogsWolves [131] dataset. In row 1 of Fig. 6.9, the first column depicts the test instance itself and columns 2 to 11 depict the 10 nearest neighbours. The second and third rows highlight the pixels that contribute the most to the similarities. The second and third rows highlight the pixels of $\boldsymbol{x}_{test}$ and $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{i10}\}$ respectively. According to row 3 of Fig. 6.9, the pink object next to the dog's leg has contributed

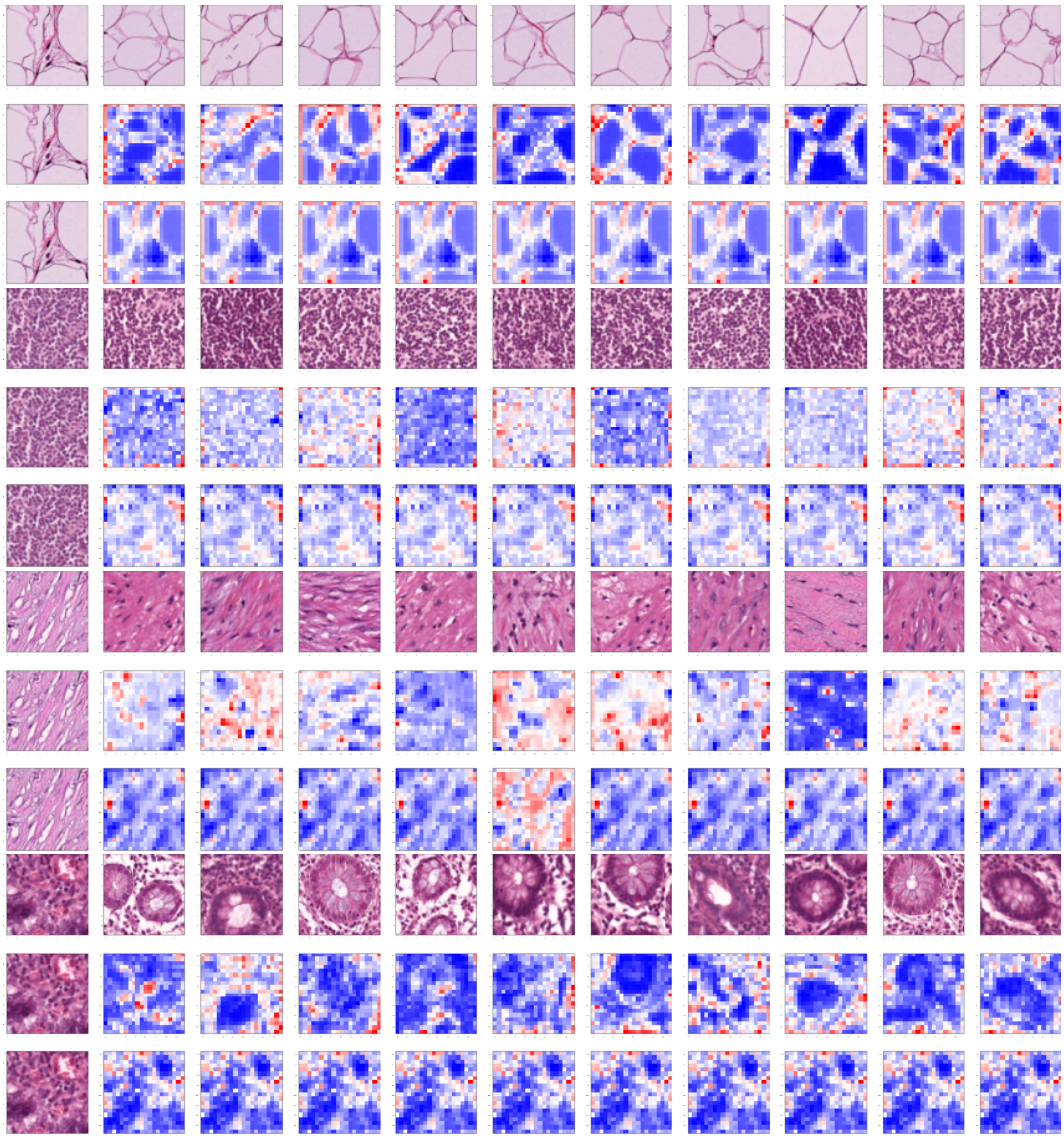


Figure 6.10: Sample explanations for Kather dataset.

the most to the similarities. According to row 2 of Fig. 6.9 regions like the baby in column 3, the dog colar or costume in columns 4, 5, and 6, human finger in column 9, and the background in columns 10 and 11 have contributed the most to their similarity to the test instance. These are patterns that usually happen for dogs images. Indeed, since the training set has been small (1600 images), to detect dogs the model is making use of patterns that normally exist in indoor scenes and do not normally appear in wolves images. We see a similar pattern for the test instance in row 6 of Fig. 6.9 and also in numerous examples in the supplementary of GPEX paper [65]. Making use of these patterns leads to misclassifications, as seen in rows 7-9 of Fig. 6.9 where the wolf is mistakenly classified as dog because of the red regions.

Fig. 6.10 illustrates some sample explanations for Kather [77] dataset. Similar to Fig. 6.9, in Fig. 6.10 rows 2, 5, 8, and 11 highlight the contributing regions of nearest neighbours while rows 3, 6, 9, and 12 highlight those of the test instance itself. According to rows 1, 2, and 3 of Fig. 6.10, the similarity is due to the wire mesh formed by cellular membranes described by our expert pathologist. In rows 4, 5, and 6 of Fig. 6.10 the test image is correctly classified as lymphocytes. For a pathologist they represent scattered well defined round structures. According to rows 4, 5, and 6 of Fig. 6.10, the model considers all regions which matches the way pathologists recognize lymphocytes. Row 7 of Fig. 6.10 shows cancer-associated stroma which is classified correctly. All 10 nearest neighbours are also cancer-associated stroma. Distinguishing between cancer-associated stroma and normal smooth muscle is a challenging task even for expert pathologists, and they often look similar. According to rows 7, 8, and 9 of Fig. 6.10, the model cares about both the stroma and nuclei. The test image in row 10 of Fig. 6.10 gets missclassified. According to rows 10, 11, and 12 of Fig. 6.10 the artificial white holes are considered as glandular lumens by the model and that can explain why the test instance gets misclassified.

Please refer to GPEX paper [65] and its supplementary for more sample interpretations.

6.4.4 Qualitative Comparision of GPEX and Representer Point Selection

We qualitatively compared the explanations of our proposed GPEX to those of representer point selection [136]. To run the method we used the publicly available code for representer point selection provided by authors in the paper [136]. The results are provided in Fig. 6.11 (and Figs. S50-S56 in the supplementary of GPEX paper [65]). In each triple, the first row shows the test instance and the 10 nearest neighbours found by our proposed GPEX. The second row shows the 10 nearest neighbours selected by representer point selection [136]. The third row shows the 10 nearest neighbours according to the kernel-space of representer point selection [136]. The formulation of representer point selection assigns an importance weight to each training instance. Therefore, some training instances tend to appear as nearest neighbours regardless of what the testing instance is. We see this behaviour in rows 2, 5, and 8 of Figs. 6.11. However, for our proposed GPEX the nearest neighbours can freely change for different test instances. We see this behaviour in rows 1, 4, and 7 of Fig. 6.11. If we ignore the importance weights in representer point selection [136], the aforementioned issue in that method happens less frequently, as we see in rows 3, 6, and 9 of Fig. 6.11. However, the issue is that without the importance weights, the explainer model in representer point selection will not be faithful to the ANN itself.

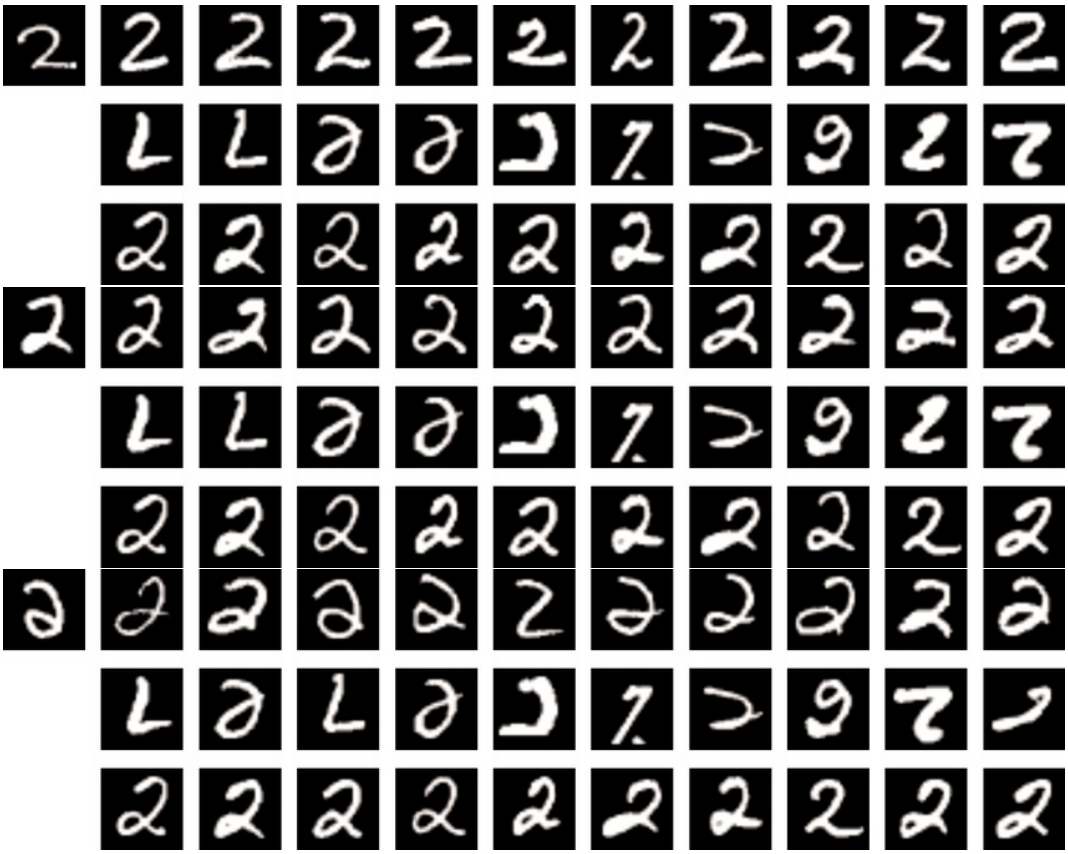


Figure 6.11: Nearest neighbours returned by the proposed GPEX vs. those returned by representer point selection [136].

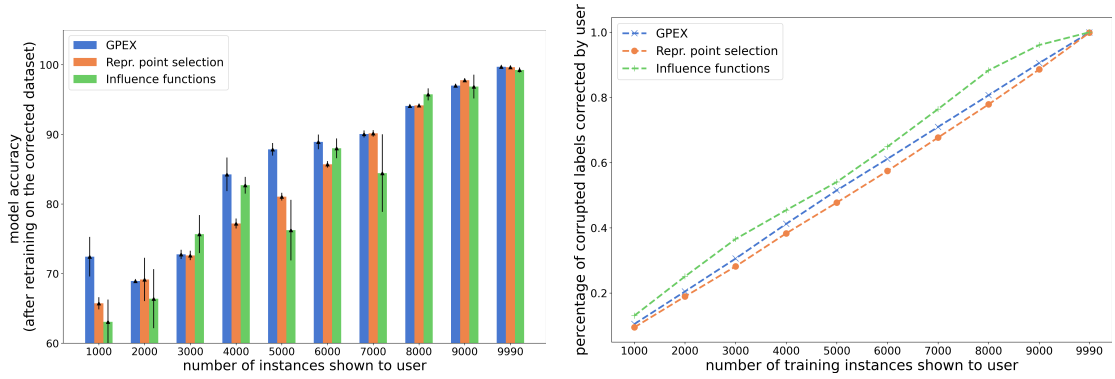


Figure 6.12: Evaluating the proposed GPEX, representer point selection [136], and influence functions [82] in dataset debugging task.

6.4.5 Evaluating GPEX in Dataset Debugging

Experimental Setup

We compared our proposed GPEX to representer point selection [136] and influence functions [82] in dataset debugging task. In these experiments we only selected images from Cifar10 [83] that are labeled as either automobile or horse. To corrupt the labels, we randomly selected 45% of training instances and changed their labels. Afterwards, we trained a classifier CNN with ResNet18 [60] backbone with the same training procedure explained in Sec. 6.4.1. In dataset debugging task, training instances are shown to a user in some order. After seeing an instance, the user checks the label of the instance and corrects it if needed. One can use explanation methods to bring the corrupted labels to the user's attention more quickly. Given an explanation method, we repeatedly select a test instance which is misclassified by the model. Afterwards, we show to the user the closest training instance (of course among the training instances which are not yet shown to the user). We repeat this process for test instances in turn until all training instances are shown to the user. We compared our proposed GPEX to representer point selection [136] and influence functions [82] in dataset debugging task. We used an implementation of influence functions [82] based on LiSSA [27] with 10 steps for each instance. The implementation is publicly available [1]. For representer point selection [136] we used the implementation by authors which is publicly available [7].

Results

The result is shown in Fig. 6.12. According to the plot on the left in Fig. 6.12, when correcting the dataset by GPEX, the model accuracy becomes close to 90% after showing about 4000 instances to user. But when using representer point selection [136] or influence functions [82], this happens when the user has seen about 7000 training instances. With noisy labels model training becomes unstable. Therefore, in the plots of Fig. 6.12 we repeat the training 5 times and we report the standard errors by the lines in top of the bars. According to the plot on the right of Fig. 6.12, after showing a fixed number of training instances to the user, when using the proposed GPEX more corrupted labels are shown to the user. Indeed, GPEX brings the corrupted labels to the user's attention quicker than representer point selection [136] does. Interestingly, according to the plot in the right hand side of Fig. 6.12 influence functions [82] is quicker at spotting incorrect labels, but the instances found by our proposed method are more effective in increasing the accuracy quicker.

6.4.6 Analysing the Effect of Number of Inducing Points

To analyze the effect of the number of inducing points (i.e. the variable M in Eq. 6.2) we applied the proposed GPEX to the classifier CNN that we trained on Cifar10 dataset [83] according to the

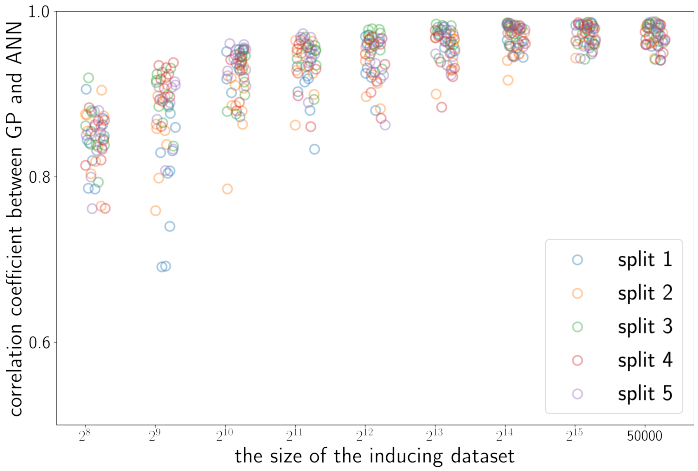


Figure 6.13: Analyzing the effect of the size of inducing dataset.

procedure of Sec. 6.4.1. But this time instead of considering all training instances as the inducing dataset, we randomly selected some training instances. In Fig. 6.13 the horizontal axis shows the size of the inducing dataset. For each size, we repeated the experiment 5 times (i.e. split 1-5 in Fig. 6.13). According to Fig. 6.13, to obtain GPs which are faithful to ANNs one needs to have a lot of inducing points. This highlights the importance of the scalability techniques that we explained in Sec. 6.3.5 to achieve a good match between GPs and ANNs. This observation is inline with the known Gaussian process behaviour when modeling a complicated function [133]. Another intriguing point in Fig. 6.13 is that if we are to select a few training images as inducing points, the correlation coefficients highly depend on which instances are selected. More precisely, Fig. 6.13 suggests that one may be able to reach high correlation coefficients by selecting a few inducing points from the training set in a subtle way.

6.5 Conclusion

In this chapter we introduced GPEX, a method for interpreting artificial neural networks. We empirically showed that this method obtains GPs which are faithful to the given ANNs, and that the kernel of the obtained GPs is highly understandable to humans and provides reliable insights about ANNs' underlying decision mechanism. Furthermore, we compared GPEX qualitatively to representer point selection [136], and quantitatively to representer point selection [136] and influence functions [82] in dataset debugging task. Finally, we showed the importance of scalability in obtaining GPs which are highly faithful to the ANNs at hand.

Chapter 7

Adopting the Proposed GPEX in Active Learning

7.1 Synopsis

In this chapter we show that in Bayesian active learning the Gaussian processes obtained by our proposed GPEX [65] are a better alternative to the commonly used dropout.

7.2 Overview

BED (Bayesian experimental design) [106] is a general framework applicable to many settings: biological experimental design [68], linear bandit problem [75], and active learning [126, 67]. The BED setting assumes that there is a ground-truth parameter $\boldsymbol{\theta}^*$ which is unknown and is to be inferred from observations. Moreover, there is a distribution $p_\theta(\boldsymbol{\theta}|\mathcal{D})$ which shows the learner's current belief about $\boldsymbol{\theta}$ after is has seen observations $\mathcal{D}$ so far. As more data is collected, $\mathcal{D}$ becomes larger and $p_\theta(\boldsymbol{\theta}|\mathcal{D})$ is ideally expected to become a Dirac delta function over $\boldsymbol{\theta}^*$. When an artificial neural network (ANN) is used as the predictive model, there is no consensus on how to model the current belief $p_\theta(\boldsymbol{\theta}|\mathcal{D})$, because ANNs are deterministic.

Dropout [127] is commonly used to model $p_\theta(\boldsymbol{\theta}|\mathcal{D})$ as follows. An ANN with dropout [127] layers is trained on the dataset $\mathcal{D}$. Afterwards, instances of the ANN with random realizations of the dropout masks are though of as samples from $p_\theta(\boldsymbol{\theta}|\mathcal{D})$. Although dropout is commonly used for this purpose, there is no theoretical backing for its adoption. More precisely, dropout [127] was created to implicitly train an ensemble of models thereby mitigating overfitting, rather than giving some weights or belief to different models of an ensemble. Some previous studies also

highlight this incapability of dropout. For example Pop *et al.* [102] showed that using an ensemble of ANNs improves the uncertainty estimates provided by Monte Carlo dropout [54]. Of note, using an ensemble of ANNs increases the computation time by a factor of ensemble size. Moreover, our experiments in Sec. 7.5 show that using an ensemble of ANNs [102] has a similar shortcoming as dropout [127] in the simulated setting. Another alternative to dropout is NTK (neural tangent kernel) [73] which has been adopted in active learning [64, 94]. But our proposed GPEX [65] is superior to NTK [73], because NTK [73] is only applicable to a multilayer perceptron whose each and every layer is wide, but our proposed GPEX [65] performs knowledge distillation regardless of the architecture of the ANN.

At the following we firstly explain how the proposed GPEX [65] can be used to model the current belief $p_\theta(\boldsymbol{\theta}|\mathcal{D})$. Afterwards, in Sec. 7.5 on simulated data we demonstrate that dropout [127] - when used to model $p_\theta(\boldsymbol{\theta}|\mathcal{D})$ - can make the active learner biased towards the decision boundary of the current predictive model, and can make the active learner completely overlook regions on which the current predictor is mistakenly certain. Finally, in Sec. 7.6 we conduct experiments on our IHC4BC dataset in active learning setting, and show that the proposed approach in Alg. 6 improves the state-of-the-art active learning method **E**xpected **P**redictive **I**nformation **G**ain (EPIG) [126].

7.3 Problem Definition

In active learning [120] a pool of instances $\mathcal{P} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_N}\}$ is available. Initially for a small subset of instances in the pool the labels are available. More precisely, initially the dataset $\mathcal{D}^{(0)} = \{(\boldsymbol{x_{i1}}, y_{i1}), ..., (\boldsymbol{x_{iN_0}}, y_{iN_0})\}$ is available, where $N_0 << N$. The task of an active learner is to ask for the labels of some unlabeled instances in the pool in a way that the newly acquired labels are the most beneficial to a machine learning method. Often there are more than one active learning cycles or rounds, and in each round the active learner can pick up a limited number of unlabeled instances in the pool and ask for their labels. Indeed, the dataset of labeled instances increases in size since the active learner asks for more labels, and we have that: $\mathcal{D}^{(0)} \subset \mathcal{D}^{(1)} \subset \mathcal{D}^{(2)} \subset ....$ Of note, in all cycles the active learner has access to the pool of unlabeled instances $\mathcal{P}$ and it may use it to, *e.g.*, estimate the distribution of instances.

In active learning there is a dilemma between exploration and exploitation, which is a essential prerequisite to comprehend the discussion of Sec. 7.5. An active learner often trains a predictor using the labels that it has seen up to that cycle. Afterwards, the active learner can make use of this predictor to decide which samples should ideally be labeled next. For example, an active learner may choose unlabeled instances on which the uncertainty of the trained predictor is the highest. All in all, an active learner faces a dilemma between exploration and exploitation[39]:

- exploitation: the predictions of the current model at hand are to some degree reliable. So if

the current predictor is quite certain about the label of an instance $\boldsymbol{x}$, the label/outcome on $\boldsymbol{x}$ is presumed known and should not be asked.

- exploration: the current model (i.e. predictor) should not be over-trusted and the label/outcome on different samples should be explored.

7.4 Proposed Method

7.4.1 Background: Active Learning for a Linear Regressor

Let $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ be a matrix containing N observed feature vectors and $\boldsymbol{y} \in \mathbb{R}^N$ be a set of continuous observed values. Also let's assume that the observations $\boldsymbol{y}$ are generated by a ground-truth and unknown linear transformation $\boldsymbol{\beta}^*$ as $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a noise vector generated from a normal distribution with zero mean and variance equal to σ. Let $\boldsymbol{\beta}$ be the linear transformation estimated from the observations and using the standard closed-form linear regression formula. We have that [116]:

$$\boldsymbol{\beta} \;\sim\; \mathcal{N}\Big(\boldsymbol{\beta}^*\,,\; \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\Big). \tag{7.1}$$

Interestingly, the variance of the estimator in Eq. 7.1 does not depend on the observations $\boldsymbol{y}$. In other words, the optimal samples that minimize the variance of the linear estimator are independent from measurements and only depend on the instances. This approach is usually referred to as classical experimental design [106] and most commonly optimizes a functional of the Fisher information matrix [106].

7.4.2 Active Learning Using GPEX

There are two views of a Gaussian process: weight-space view and function-space view [107]. These two views are equivalent [107]. More specifically, let $f : \mathbb{R}^D \rightarrow \mathbb{R}^{D_u}$ be a function that maps an instance to a kernel space as: $\boldsymbol{u} = f(\boldsymbol{x})$. Moreover, let $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{u}_i^T \boldsymbol{u}_j = f(\boldsymbol{x}_i)^T f(\boldsymbol{x}_j)$ be the kernel function of a Gaussian process. It is easy to show that the mean of the GP's posterior distribution on any $\boldsymbol{x}_*$ is equal to $\boldsymbol{\beta}^T f(\boldsymbol{x}_*)$, where $\boldsymbol{\beta} \in \mathbb{R}^{D_u}$ is the linear transformation estimated using the standard closed-form linear regression formula and from the observations in the space of the $\boldsymbol{u}$ vectors in $\mathbb{R}^{D_u}$. Intuitively, the GP is equivalent to a linear regression in the space of the $\boldsymbol{u}$ vectors. We used this notion to propose Alg. 6. In Alg. 6 a Gaussian process is matched to the given neural network (line 1 of Alg. 6). Afterwards, random linear transformations are generated in the kernel-space (line 9 of Alg. 6). According to line 10, the generated random functions are the kernel-space mapping $f(.)$ followed by the random linear transformations. Of note, in line 1 of Alg. 6 the GP is fitted to the neural network on both labeled and unlabeled samples of the pool, and

doing so is valid because the active learner has access to both labeled and unlabeled instances in the pool.

Algorithm 6 GPEX's part in Bayesian active learning

Input:

$\mathcal{P}$, Pool of labeled and unlabeled instances.

$\mathcal{D}$, Dataset of labeled instances.

$g(.)$, A neural network which is trained on $\mathcal{D}$.

S, The number of generated and returned random functions.

Output:

$[\hat{g}_1(.), \hat{g}_2(.), ..., \hat{g}_S(.)]$ List of functions to be considered as samples from the current belief about the unknown parameter, *i.e.* samples from $p(\boldsymbol{\theta}|\mathcal{D})$ in Bayesian experimental design.

$GP \leftarrow GPEX\big(g(.), \mathcal{P}\big)$ ▷ Use GPEX [65] to find a GP that matches $g(.)$ on pool instances $\mathcal{P}$.
$f(.) \leftarrow GP.kernel\_module$ ▷ Consider the kernel mapping of the obtained GP.
for $n = 1 \rightarrow |\mathcal{D}|$ **do** ▷ Transform the labeled instances by $f(.)$.
    $\boldsymbol{u_n} \leftarrow f(\boldsymbol{x_n})$
end for
$\boldsymbol{U_{obs}} \overset{\text{def}}{=} [\boldsymbol{u_1}, ..., \boldsymbol{u_{|\mathcal{D}|}}]$ and $\boldsymbol{y} = [y_1, y_2, ..., y_{|\mathcal{D}|}]$
$output = []$
for $s = 1 \rightarrow S$ **do**
    $\boldsymbol{\beta_s} \sim \mathcal{N}\big((\boldsymbol{U_{obs}}^T \boldsymbol{U_{obs}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{U_{obs}}^T \boldsymbol{y}\ ,\ \sigma^2 (\boldsymbol{U_{obs}}^T \boldsymbol{U_{obs}})^{-1}\big)$
    $output.append\big(\boldsymbol{\beta_s}^T\ f(.)\big)$ ▷ The generated function is $f(.)$ followed by $\boldsymbol{\beta_s}$
end for
return *output*

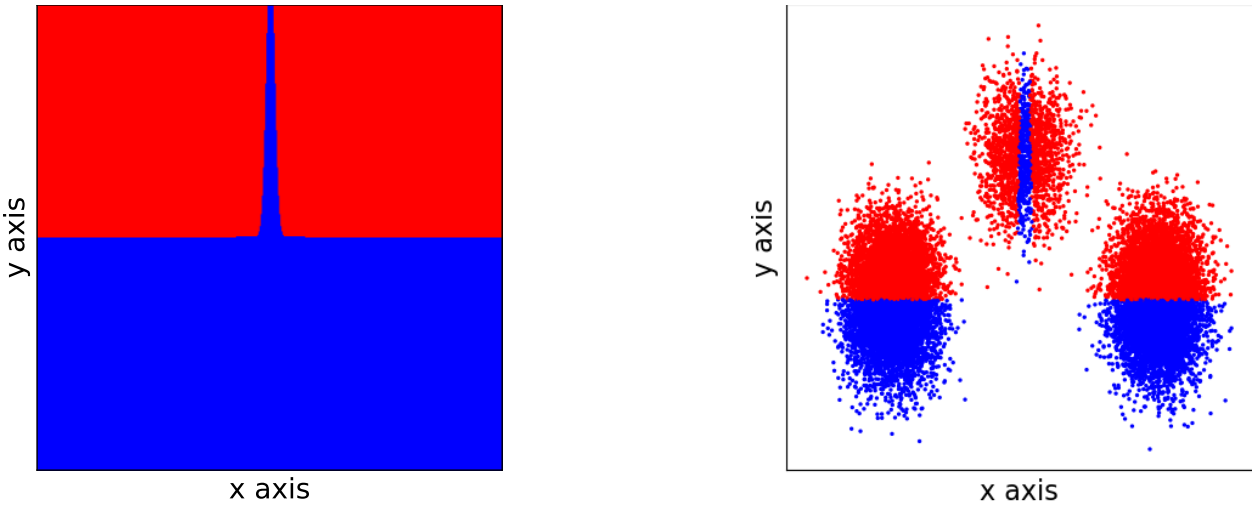
7.5 Experiments on Simulated Data

7.5.1 Simulated Dataset (Sundog)

The synsthetic dataset, which we refer to as "sundog", contains 2-dimensional instances. As shown in Fig. 7.1a the decision boundary is $y = 0.5 \cdot exp(-x^2/0.01)$ that divides the 2-dimensional space to red regions (class 1) and blue regions (class 0). The instances are generated from three Gaussian distributions, positioned as seen in Fig. 7.1b. In Fig. 7.1b the decision boundary crosses the two cluster on the left and right in horizontal direction. But the decision boundary crosses the middle cluster twice and almost vertically.

7.5.2 Inspecting Active Learning Scores on Simulated Data

We generated a simulated dataset with 1000 instances with the clusters positioned as in Fig. 7.1b. Afterwards we considered an ANN (here a multi-layer perceptron) with 3 linear layers with 100,



(a) (b)

Figure 7.1: The simulated dataset. a) the decision boundary splits the 2-dimensional space to red regions (class 1) and blue regions (class 0). b) Instances are generated from three Gaussian distribution.

50, and 2 (for 2 classes) neurons. In each layer we placed a dropout [127] layer followed by ReLU activation. We masked the middle-cluster, and trained the ANN for 1000 iterations with and Adam [80] optimizer and learning rate of 0.001. Since the cluster in the middle is masked from the ANN during training, the ANN would understandably learn a horizontal decision boundary. An active learning method should ideally pick the unseen instances in the middle cluster. But here we show that random ANNs generated by either dropout [127] or ANN-ensemble [102] all have a horizontal decision boundary.

In Fig. 7.2 the white points are instances of the simulated dataset, the heatmaps show the difference between output heads of each neural network. Moreover, the black lines roughly show the decision boundary, *i.e.* points where the difference of ANN's output heads are close to zero. We trained 5 ANNs (*i.e.* ensemble of ANNs) which are shown in the 1st row of Fig. 7.2. We see that all of these ANNs roughly have a horizontal boundary. Afterwards, we trained ANNs with different dropout probabilities: 0.2 (2nd row of Fig. 7.2), 0.4 (3rd row of Fig. 7.2), 0.6 (4th row of Fig. 7.2), and 0.8 (5th row of Fig. 7.2). Subsequently, we showed each ANN's output for 5 random dropout mask (rows 2-5 in Fig. 7.2). In Fig. 7.2 we observe that the ANNs generated by either an ensemble or dropout have a horizontal decision boundary.

To demonstrate how this can be an issue in the active learning setting, we computed the score given by the state-of-the-art active learning method EPIG [126] and with different dropout values: 0.2 (Fig. 7.3a), 0.4 (Fig. 7.3b), 0.6 (Fig. 7.3c), and 0.8 (Fig. 7.3d). To compute the EPIG scores [126] at each x-y position we used 50 random ANNs (*i.e.* dropout masks). Fig. 7.3 demonstrates

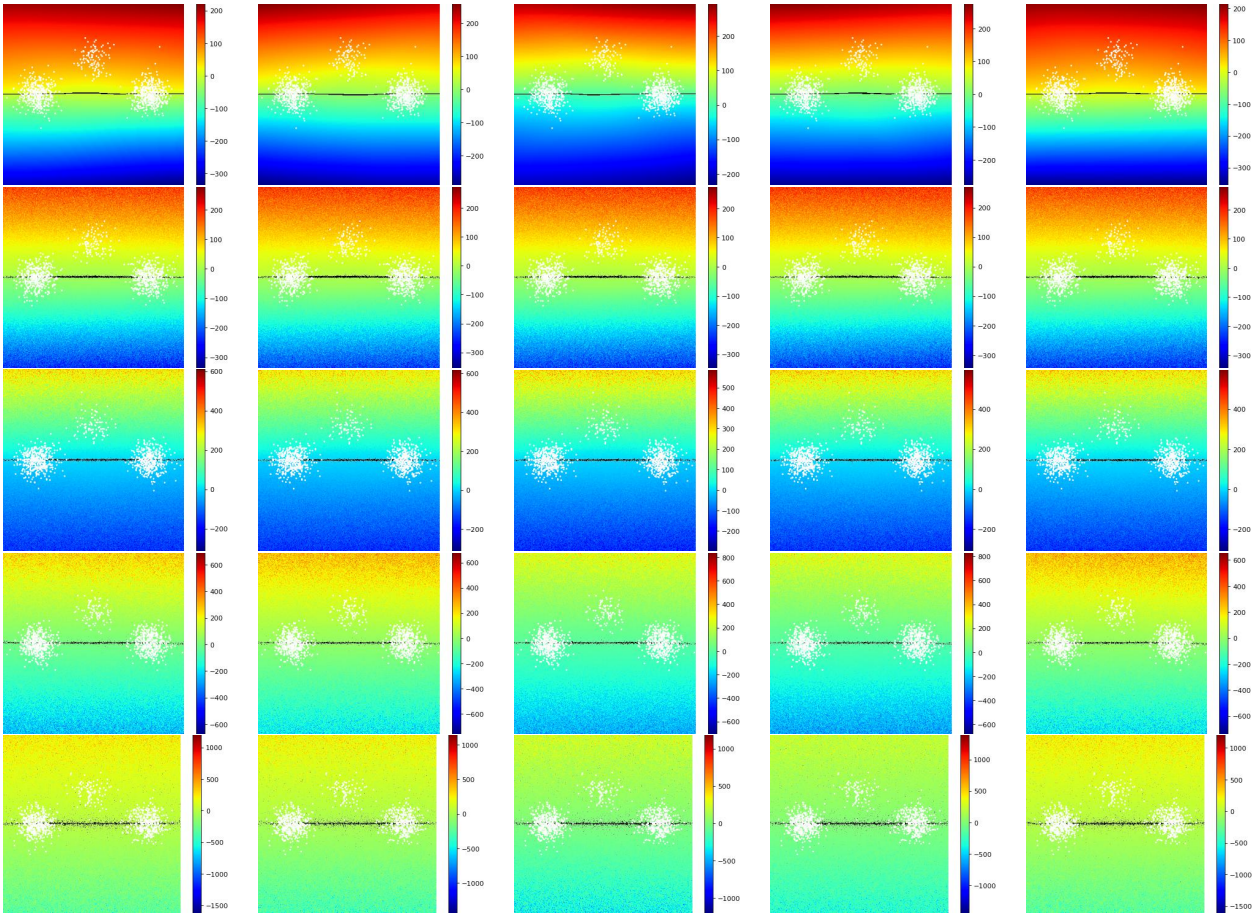
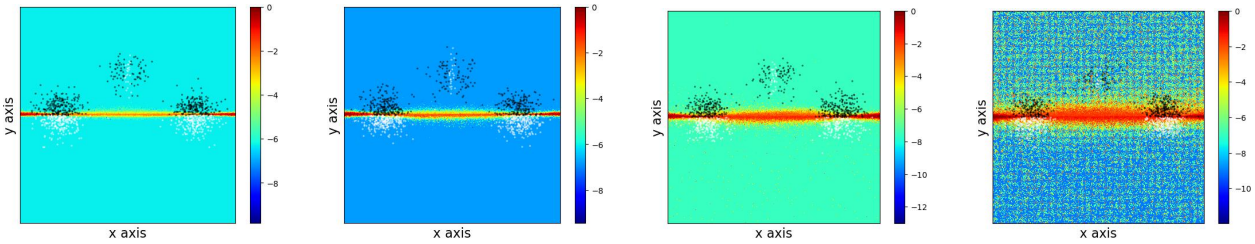


Figure 7.2: In these plots the white points are instances of the simulated dataset. Moreover, the heatmaps show the difference between output heads of each neural network, and the black line roughly shows the decision boundary, *i.e.* points where the difference of ANN's output heads are close to zero. Row 1: An ensemble of 5 ANNs trained on the simulated dataset. Rows 2-5: outputs from a trained ANN with different dropout masks, when dropout probability is set to 0.2 (2nd row), 0.4 (3rd row), 0.6 (4th row), and 0.8 (5th row).



(a) dropout prob. 0.2 (b) dropout prob. 0.4 (c) dropout prob. 0.6 (d) dropout prob. 0.8

Figure 7.3: Active learning scores computed by EPIG [126] and dropout, after min-max normalization and in log-scale.

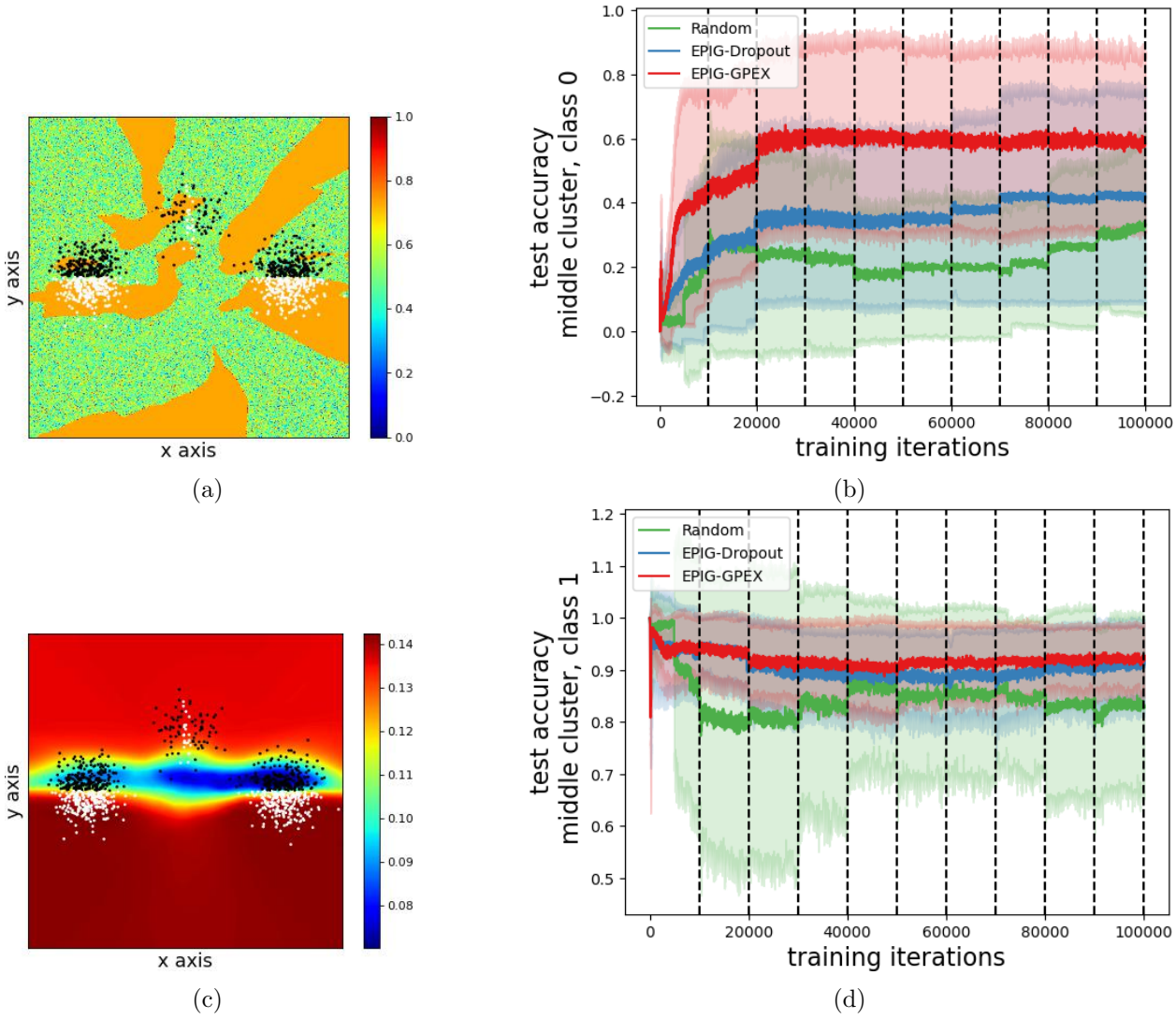


Figure 7.4: Active learning results on sundog simulated dataset. a) The heatmap shows scores computed by EPIG [126] and the proposed Alg. 6. c) The heatmap shows the uncertainy of the GP obtained in Alg. 6. b,d) test accuracy in the active learning setting explained in Sec. 7.5.3 in each training step and averaged over 10 repetitions. The vertical dashed lines separate active learning cycles Plots in b and c show the test accuracy among instances with ground-truth class 0 and 1, respectively.

that even when dropout probability is as high as 0.8 the scores computed by EPIG [126] are highly biased to the decision boundary of the ANN. In other words, according to Fig. 7.3 EPIG [126] - when used along with dropout [127] - considers only pool instances which are close to the ANN's decision boundary, and ignores the rest of pool instances. But this issue doesn't happen with the proposed Alg. 6. To demonstrate this, we computed EPIG [126] scores with the exact same parameters, but this time instead of dropout we used Alg. 6. The result is provided in Fig. 7.4a.

We see that the score heatmap in Fig. 7.4a is exploratory, unlike the scores in Fig. 7.3 which are biased to the current decision boundary.

7.5.3 Active Learning on Simulated Data (Sundog)

We repeated the following experiment 10 times. A sundog dataset with 1000 instances and with the following specs was generated: left, middle, and right modes were located at $[-2.5, 0.0]$, $[0.0, 1.5]$, and $[2.5, 0.0]$ and had probabilities 0.4, 0.2, and 0.4, respectively. Moreover, the standard variation of all Gaussian modes was set to 0.4. Initially 500 instances were randomly selected from left and right clusters (and not the middle cluster). Afterwards, 10 cycles of active learning were performed and in each cycle the active learner was asked to pick 10 unlabeled instances from the unlabeled pool. Initially each active learner was provided with an ANN with 3 linear layers with 100, 50, and 2 (for 2 classes) neurons and with dropout (probability 0.2) and ReLU activations. In each cycle each active learner picked 10 instances and the ANN was trained for 10K iterations from the previous checkpoint and with an Adam optimizer with learning rate 0.001.

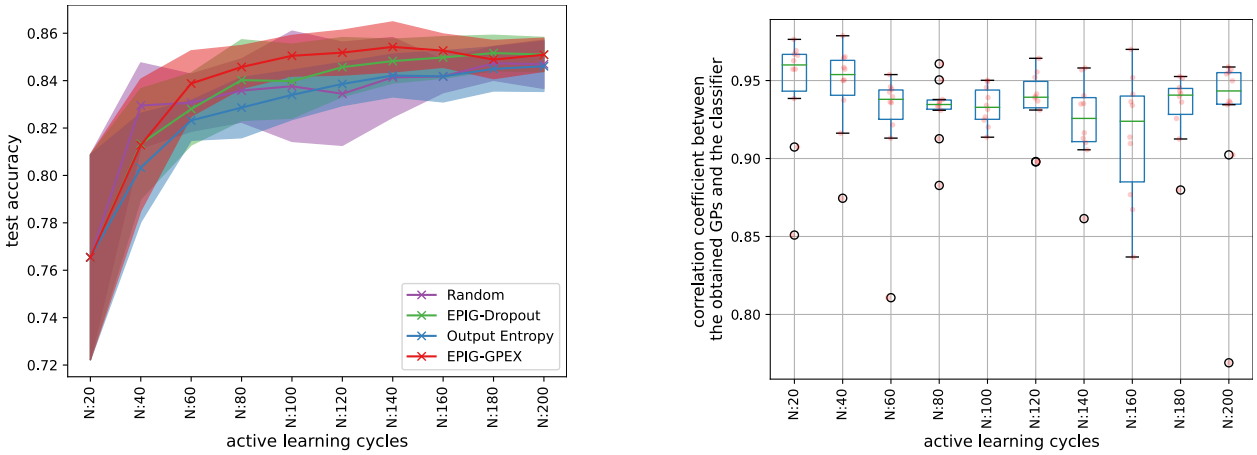
Test accuracy on the middle cluster and among class 0 (blue) and class 1 (red) instances are provided in Figs. 7.4b and 7.4d respectively. In this setting, the middle cluster in Fig. 7.1b has been unseen to the ANN during training, and therefore the ANN presumes that the middle cluster belongs to class 1. According to Fig. 7.4b the proposed Alg. 6 selects those instances sooner, and finally some of the blue instances in the middle cluster get classified as class 0. As seen in Fig. 7.4b this happens at some point for the other two baselines, but with the proposed Alg. 6 it happens way sooner. Interestingly, in Figs. 7.4b and 7.4d both versions of EPIG [126] outperform the random baseline.

7.6 Experiments on IHC4BC HER2 Dataset

7.6.1 A Patch Dataset Created from Manual Annotations

We evaluated the proposed Alg. 6 on a subset of our IHC4BC HER2 dataset [30, 6].The HER2 subset contains images from 52 patients. We randomly selected 36 patients for training and the rest (16 patients) for testing. In the training and testing sets there were 15 and 6 patients with some HER2 3+ regions, respectively. Afterwards, we used the manually annotated spots that we introduced in Sec. 5.5.1 to obtain 224 by 224 patches and their corresponding labels as follows:

- If a 3K by 3K image is labeled as negative in the original IHC4BC dataset, we assume that there is no 3+ region on the image. Therefore any 224 by 224 patch extracted from that 3K by 3K image is devoid of 3+ HER2 regions and is labeled as 0. In sum, in this case given the



(a) (b)

Figure 7.5: Active learning results on a subsmapled dataset from IHC4BC Her2 dataset.

3K by 3K patch we extracted a 224 by 224 patch from a random pixel position and labeled it as 0.

- If a spot is manually marked on a 3K by 3K image, we assume that there is a 3+ HER2 region in the vicinity of the spot. Therefore, we add a random shift of ± 50 pixels on that spot, extract a 224 by 224 patch from that pixel position, and label it as 1.

For each 3K by 3K image in IH4BC [30] HER2 dataset we obtained one patch and its corresponding label using the aforementioned procedure. Afterwards, each 224 by 224 patch was normalized with Imagenet [51] normalization parameters and was fed to a ResNet-18 backbone [60] pretrained on Imagenet [51] to obtain a 512-dimensional vector for each image.

We repeated the following active learning setup 10 times. An initial labeled pool containing 10 positive and 10 negative patches was considered. Afterwards an ANN was trained on the extracted 512-dimensional features. The ANN has 4 layers with 200, 20, 1024, and 2 (output head) neurons with ReLU activations and dropout [127] with masking probability 0.5. Different active learners were provided with this initial model, and were asked to select 20 instances from the pool and for 10 active learning cycles. In each cycle the ANN was trained for 1000 iterations, and in the last 4 iterations the validation accuracy was computed and the best checkpoint was picked as the best model in that cycle. We used an Adam [80] optimizer with learning rate 0.001. Like previous section, in each cycle the best model and its corresponding optimizer are saved and used as the initial model and checkpoint in the next cycle. To further prevent overfitting we trained the ANNs with an ℓ_2 regularization on weights with regularization coefficient 0.0001.

We evaluated 4 active learning policies using the setting mentioned above: 1. random, 2. network output entropy, 3. EPIG [126] along with dropout [127], 4. EPIG [126] along with the proposed Alg.

6. In Fig. 7.5a these four methods are labeled as "Random", "Output Entropy", "EPIG-Dropout", and "EPIG-GPEX", respectively. According to Fig. 7.5a the proposed Alg. 6 outperforms the other 3 methods, and provides more stable improvements. The benefit of using Gaussian processes in active learning was also reported in previous studies, for example in Mohamadi *et al.* [94] and Holzmüller *et al.* [64].

7.6.2 A Huge Pool of HER2 Whole-Slide Images

The experiments of Sec. 7.6.1 demonstrated the usability of GPEX for a small subset of the IHC4BC dataset. In this section we examine if GPEX can be adopted for the real clinical scenario where a HER2 predictor is trained on the IHC4BC dataset, and the goal is to optimally choose new training instances from a pool of WSIs. In this section we present two findings:

- In this setting, finding highly-faithful GPs is challenging due to the huge size of the dataset (300 GB). Notably, recall from Sec. 6.4.6 that GP's inducing dataset should be large and diverse. In this section we show that adopting GPEX is still feasible even in this challenging setting.
- In active learning we show that Alg. 6 is preferable to using GP's uncertainty, although GP's uncertainty is a popular and widely-accepted choice for active learning [76, 119, 116, 91].

Experimental Setup

An important challenge was to include a diverse set of patches as GP's inducing dataset. To this end, we extracted 120 by 120 negative patches from IHC4BC HER2 partition, fed them to a ResNet backbone, and applied kMeans clustering with 17 clusters to the extracted features There was no tangible increase in Silhouette score with more than 17 clusters. Afterwards, we tried to include in the inducing dataset the same number of patches from different cluster thereby encouraging diversity. To include HER2 positive patches, we extracted 120 by 120 patches with the protocol of Sec. 5.5.1 and subsampled them. All in all 143,443 images of size 120 by 120 were included in GP's inducing dataset. To evaluate the success of knowledge distillation, we selected 895 WSIs which were never included in the IHC4BC dataset and extracted 30 patches of size 1000 by 1000 from each WSI and from Otsu's [99] foreground region. Finally, each 1000 by 1000 patch was observed with a window size of 120 and stride of 500, and the GP-ANN match was evaluated. A strongly-supervised HER2 predictor was trained according to the procedure of Sec. 5.5.2. Notably, validation set was created according to the procedure of Sec. 5.5.2, but unlike Sec. 5.5.2 no patient was held out for testing because here for testing we used 188 whole-slide images which aren't included in the IHC4BC dataset. We applied GPEX with the same setup as Sec. 6.4. Since we have a binary

classification setting, we used an classifier with a single output head and therefore used a single GP and kernel-space.

Results

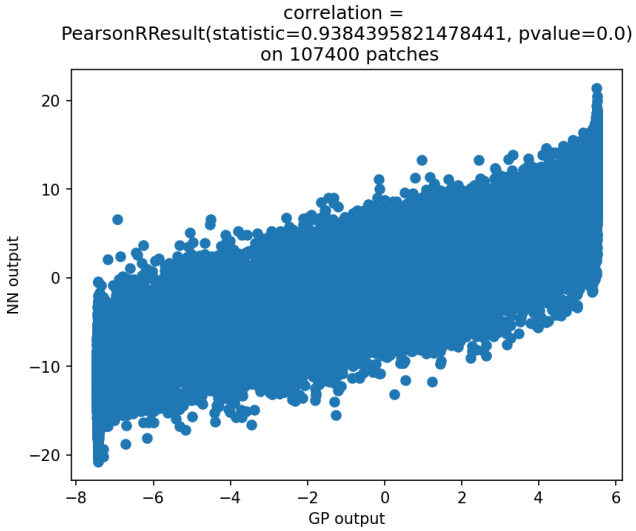
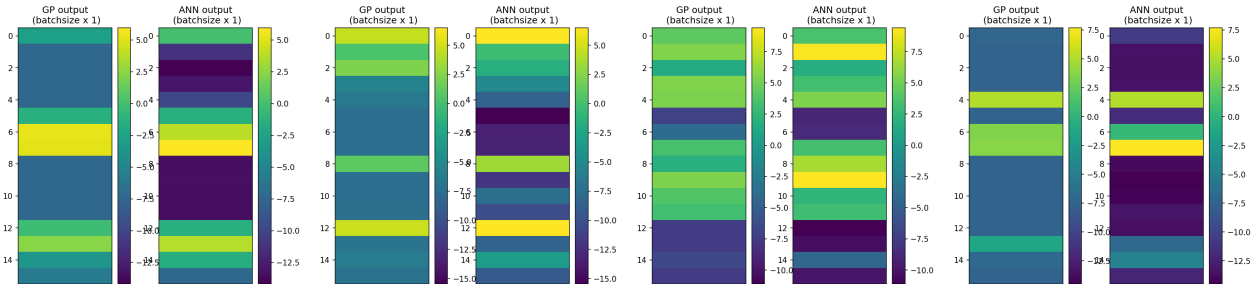


Figure 7.6: Scatter plot of GP's posterior mean versus the output of the predictor ANN.



minibatch 1 minibatch 2 minibatch 3 minibatch 4

Figure 7.7: Comparing GP and ANN outputs for four batches of images to evaluate the GP-ANN match discussed in Sec. 7.6.2.

We obtained a GP whose posterior mean closely matches the output of the ANN. According to Fig. 7.6 when one uses 150K inducing points, the correlation coefficient is around 0.9384, which is illustrated in Fig. 7.7 the GP and ANN are reasonably close. We tried to find equivalent GPs with fewer inducing points than 150K (roughly 10K, 20K, 30K, 40K, and 50K) and interestingly none of those experiments were successful. This highlights the challenge of applying GPEX to such a huge dataset, and corroborates the results of Sec. 6.4.6.

We inspected the predictor and the obtained GP on 188 WSIs set aside for testing. The first

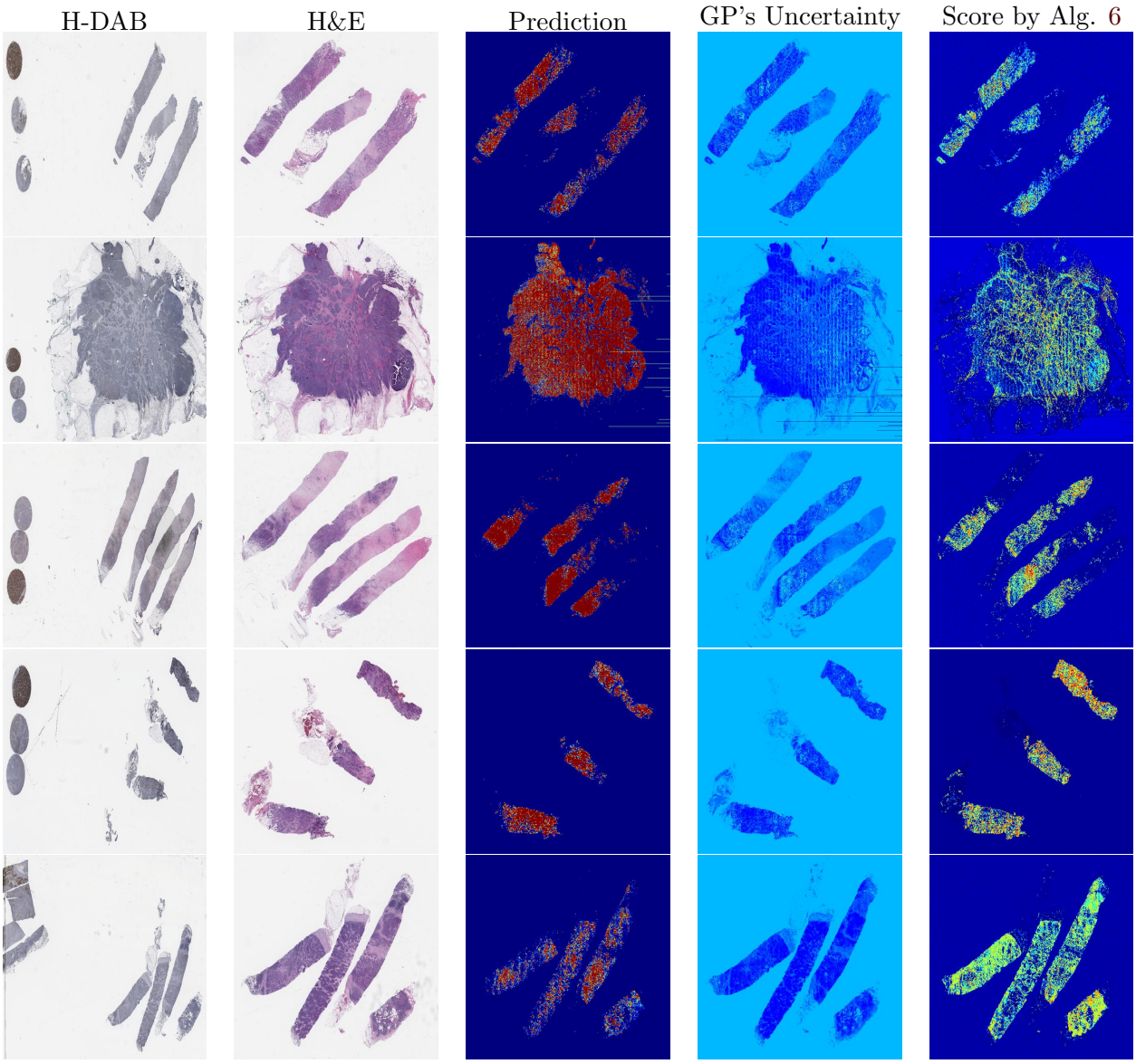


Figure 7.8: The HER2 predictor and different active learning strategies applied to 5 sample WSIs.

observation was that the majority of failures are false positives usually for 1+ and occasionally for negative cases. Five such cases are shown in Fig. 7.8 where in the third column the prediction is mistakenly positive. An important question is: can an active learning method automatically detect this defect by giving a high score to such regions? According to the forth column of Fig. 7.8 the uncertainty of GP (a popular active learning strategy) mistakenly gives a low score to such regions, and gives more score to the white background. This negative result is important, because many works take for granted that GP's uncertainty is a good active learning strategy [76, 119, 116, 91]. Nonetheless, in the fifth columns of Fig. 7.8 we observe that EPIG [126] along with GPEX correctly

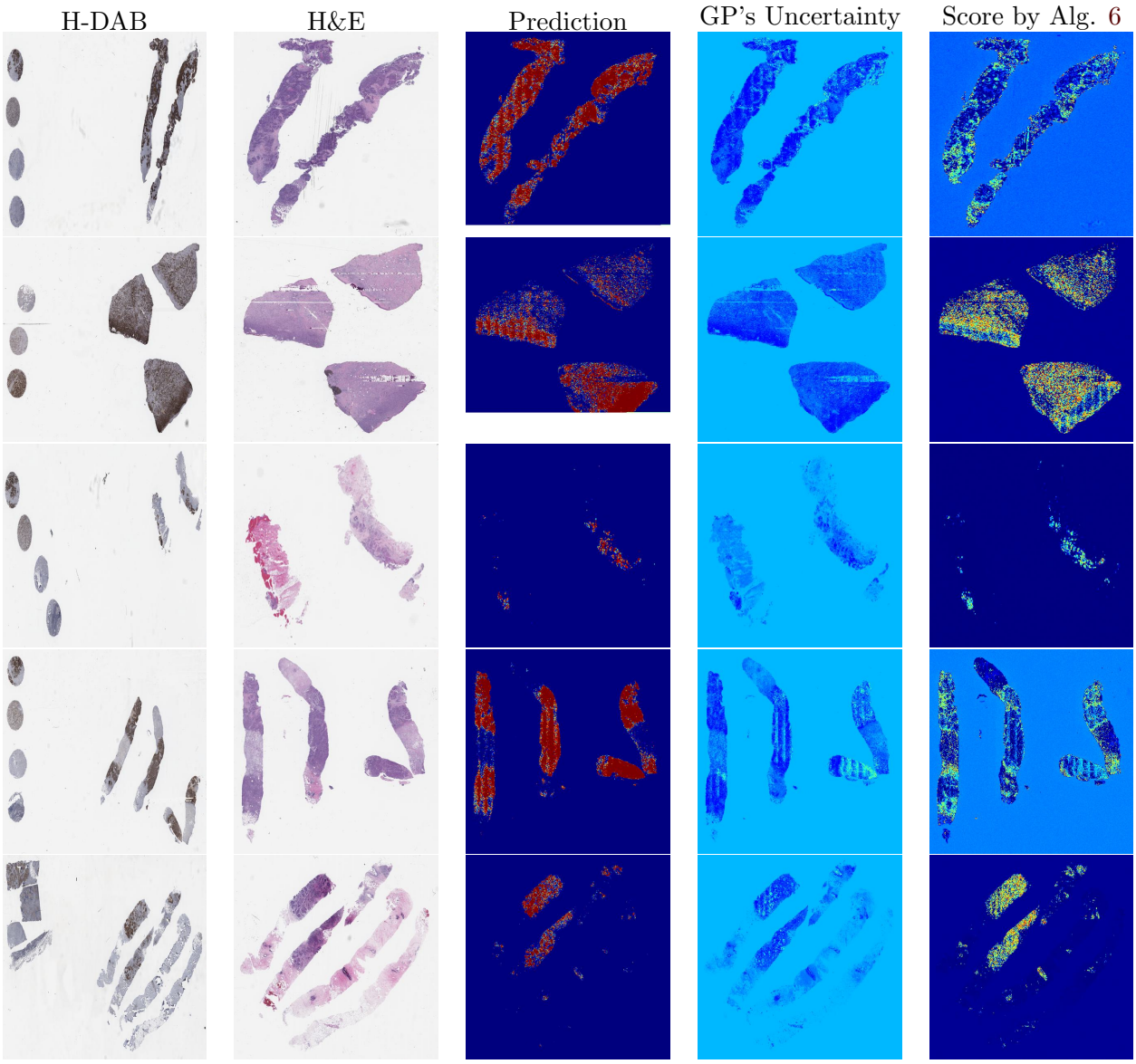


Figure 7.9: The HER2 predictor and different active learning strategies applied to 5 sample WSIs.

highlights such regions. Interestingly, according to third column of Fig. 7.8 the output activations for the false positive regions are quite high and close to 1.0. Therefore, output entropy as an active learning strategy has no way to give a high score to such regions. Finally, we observed that most 3+ regions are correctly identified by the predictor, as seen in Fig. 7.9.

7.7 Conclusion

In this chapter we conducted active learning experiments on simulated data and a small subset of the IHC4BC HER2 dataset, and showed that for active learning our proposed GPEX is a better alternative to the commonly used dropout [127]. Furthermore, we conducted experiments on the entire IHC4BC HER2 dataset and showed that EPIG [126] along with GPEX is preferable to two popular active learning strategies: output entropy and GP's uncertainty. As importantly, we showed that adopting GPEX is still feasible for such a huge dataset of size 300 GB.

References

[1] A simple PyTorch implementation of influence functions. https://github.com/alstonlo/torch-influence. [Online; accessed 19-Oct-2023].

[2] An implementation for ResNet in pytorch. https://github.com/kuangliu/pytorch-cifar. [Online; accessed 19-Dec-2021].

[3] The cancer genome atlas. portal.gdc.cancer.gov. Accessed: 2020-09-22.

[4] GPEX github repository, . https://github.com/amirakbarnejad/gpex. [Online; accessed 23-March-2024].

[5] HistoVAE repository, . https://github.com/willgdjones/HistoVAE/. [Online; accessed 23-March-2024].

[6] IHC4BC dataset webpage, . https://ihc4bc.github.io/. [Online; accessed 23-March-2024].

[7] Implementaiton of represnter point selection. https://github.com/chihkuanyeh/Representer_Point_Selection. [Online; accessed 19-Oct-2023].

[8] An implementation for vision transformers. https://towardsdatascience.com/implementing-visualttransformer-in-pytorch-184f9f16f632 . Accessed: 2010-09-30.

[9] Implementation of clustering-constrained attention multiple instance learning (clam). https://github.com/mahmoodlab/CLAM . Accessed: 2010-09-30.

[10] Implementation of the Proposed method based in Fisher vector encoding, . https://github.com/amirakbarnejad/code_submission_isbi2021/tree/main. [Online; accessed 23-March-2024].

[11] Increasing batch-size by gradient accumulation, . https://kozodoi.me/blog/20210219/gradient-accumulation. [Online; accessed 23-March-2024].

[12] Karen stiffler, what is histotechnology? https://www.lakelandcc.edu/c/document_library/get_file?uuid=6b307b80-d0ab-4056-a744-ea10768a3062&groupId=357824. Accessed: 2023-07-10.

[13] mil, a multiple instance learning library for Python, . https://github.com/rosasalberto/mil. [Online; accessed 30-March-2024].

[14] Online Documentation for GPEX, . https://gpex.readthedocs.io/en/latest/. [Online; accessed 23-March-2024].

[15] Webpage. https://www.dfhcc.harvard.edu/research/core-facilities/rodent-histopathology/. Accessed: 2023-07-10.

[16] Webpage. https://en.wikipedia.org/wiki/H%26E_stain. Accessed: 2023-07-10.

[17] Webpage. https://en.wikipedia.org/wiki/Antibody. Accessed: 2023-07-11.

[18] Webpage. https://www.enzolifesciences.com/science-center/technotes/2019/august/what-are-the-different-detection-methods-for-ihc?/. Accessed: 2023-07-11.

[19] Webpage. https://en.wikipedia.org/wiki/Fluorescence_microscope. Accessed: 2023-07-11.

[20] Webpage. https://camelyon17.grand-challenge.org/Background/. Accessed: 2023-07-16.

[21] Webpage. https://www.leicabiosystems.com/en-ca/knowledge-pathway/an-introduction-to-specimen-processing/. Accessed: 2023-07-10.

[22] Webpage. https://en.wikipedia.org/wiki/Fluorescence_in_situ_hybridization. Accessed: 2023-07-11.

[23] Webpage. https://www.accessdata.fda.gov/cdrh_docs/reviews/K172174.pdf. Accessed: 2023-07-16.

[24] Webpage. https://en.wikipedia.org/wiki/Digital_pathology. Accessed: 2023-07-16.

[25] Webpage. https://en.wikipedia.org/wiki/Virtual_microscopy. Accessed: 2023-07-16.

[26] Webpage. https://github.com/amirakbarnejad/PyDmed. Accessed: 2023-07-16.

[27] Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research 18*, 1 (2017), 4148–4187.

[28] Agarwal, S., Jabbari, S., Agarwal, C., Upadhyay, S., Wu, S., and Lakkaraju, H. Towards the unification and robustness of perturbation and gradient based explanations. In *International Conference on Machine Learning* (2021), PMLR, pp. 110–119.

[29] Agresti, A. Categorical data analysis. hoboken, 2002.

[30] Akbarnejad, A., Ray, N., Barnes, P. J., and Bigras, G. Predicting ki67, er, pr, and her2 statuses from h&e-stained breast cancer images. *arXiv preprint arXiv:2308.01982* (2023).

[31] Akbarnejad, A., Ray, N., and Bigras, G. Deep fisher vector coding for whole slide image classification. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)* (2021), IEEE, pp. 243–246.

[32] Akbarnejad, A. H., and Baghshah, M. S. An efficient semi-supervised multi-label classifier capable of handling missing labels. *IEEE Transactions on Knowledge and Data Engineering 31*, 2 (2019), 229–242.

[33] Anand, D., Kurian, N. C., Dhage, S., Kumar, N., Rane, S., Gann, P. H., and Sethi, A. Deep learning to estimate human epidermal growth factor receptor 2 status from hematoxylin and eosin-stained breast tissue images. *Journal of pathology informatics 11*, 1 (2020), 19.

[34] Arun et al., K. S. Enhanced bag of visual words representations for content based image retrieval: a comparative study. *Artificial Intelligence Review 53*, 3 (Mar 2020), 1615–1653.

[35] Avanti Shrikumar et al. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning* (06–11 Aug 2017), vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 3145–3153.

[36] Awan, R., Koohbanani, N. A., Shaban, M., Lisowska, A., and Rajpoot, N. Context-aware learning using transferable features for classification of breast cancer histology images. In *Image Analysis and Recognition: 15th International Conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15* (2018), Springer, pp. 788–795.

[37] Babenko, B. Multiple instance learning: algorithms and applications. *View Article PubMed/NCBI Google Scholar 19* (2008).

[38] Borup, K., and Andersen, L. N. Self-distillation for gaussian process regression and classification. *arXiv preprint arXiv:2304.02641* (2023).

[39] Bouneffouf, D., Laroche, R., Urvoy, T., Féraud, R., and Allesiardo, R. Contextual bandit for active learning: Active thompson sampling. In *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I 21* (2014), Springer, pp. 405–412.

[40] Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D. A., Chang, J. H., Lindquist, R. A., Moffat, J., et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology 7* (2006), 1–11.

[41] Ceccarelli, M., Barthel, F. P., Malta, T. M., Sabedot, T. S., and Salama et al., S. R. Molecular profiling reveals biologically discrete subsets and pathways of progression in diffuse glioma. *Cell 164*, 3 (Jan 2016), 550–563. 26824661[pmid].

[42] Chen, H., Han, X., Fan, X., Lou, X., Liu, H., Huang, J., and Yao, J. Rectified cross-entropy and upper transition loss for weakly supervised whole slide image classifier. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part I 22* (2019), Springer, pp. 351–359.

[43] Chen, N., and Zhou, Q. The evolving gleason grading system. *Chinese Journal of Cancer Research 28*, 1 (2016), 58.

[44] Chen, R. J., Chen, C., Li, Y., Chen, T. Y., Trister, A. D., Krishnan, R. G., and Mahmood, F. Scaling vision transformers to gigapixel images via hierarchical self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 16144–16155.

[45] Cifci, D., Foersch, S., and Kather, J. N. Artificial intelligence to identify genetic alterations in conventional histopathology. *The Journal of Pathology 257*, 4 (2022), 430–444.

[46] Ciga, O., Xu, T., and Martel, A. L. Self supervised contrastive learning for digital histopathology. *Machine Learning with Applications 7* (2022), 100198.

[47] Cohen, M. K., Daulton, S., and Osborne, M. A. Log-linear-time gaussian processes using binary tree kernels. In *Advances in Neural Information Processing Systems* (2022), A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds.

[48] Comiter, C., Vaishnav, E. D., Ciampricotti, M., Li, B., Yang, Y., Rodig, S. J., Turner, M., Pfaff, K. L., Jané-Valbuena, J., Slyper, M., et al. Inference of single cell profiles from histology stains with the single-cell omics from histology analysis framework (schaf). *BioRxiv* (2023), 2023–03.

[49] Conde-Sousa, E., Vale, J., Feng, M., Xu, K., Wang, Y., Della Mea, V., La Barbera, D., Montahaei, E., Baghshah, M., Turzynski, A., Gildenblat, J., Klaiman, E., Hong, Y., Aresta, G., Araújo, T., Aguiar, P., Eloy, C., and Polónia, A. Herohe

challenge: Predicting her2 status in breast cancer from hematoxylin and eosin whole-slide imaging. *Journal of Imaging 8*, 8 (2022).

[50] de G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations* (2018).

[51] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[52] Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine 29*, 6 (2012), 141–142.

[53] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[54] Gal, Y., and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (2016), PMLR, pp. 1050–1059.

[55] Ghaffari Laleh, N., Ligero, M., Perez-Lopez, R., and Kather, J. N. Facts and hopes on the use of artificial intelligence for predictive immunotherapy biomarkers in cancer. *Clinical Cancer Research 29*, 2 (2023), 316–323.

[56] Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (Jul. 2019), 3681–3688.

[57] Guruprasad, N., Akbarnejad, A., Bigras, G., Barnes, P. J., and Ray, N. A closer look at weak supervision's limitations in wsi recurrence score prediction. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2023), pp. 1941–1946.

[58] Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems 31* (2018).

[59] He, B., Bergenstråhle, L., Stenbeck, L., Abid, A., Andersson, A., Borg, Å., Maaskola, J., Lundeberg, J., and Zou, J. Integrating spatial gene expression and breast tumour morphology via deep learning. *Nature biomedical engineering 4*, 8 (2020), 827–834.

[60] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[61] He, L., Ray, N., Guan, Y., and Zhang, H. Fast large-scale spectral clustering via explicit feature mapping. *IEEE Transactions on Cybernetics 49*, 3 (2019), 1058–1071.

[62] Henaff, O. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 4182–4192.

[63] Höhne, J., de Zoete, J., Schmitz, A. A., Bal, T., di Tomaso, E., and Lenga, M. Detecting genetic alterations in braf and ntrk as oncogenic drivers in digital pathology images: Towards model generalization within and across multiple thyroid cohorts. In *MICCAI Workshop on Computational Pathology* (2021), PMLR, pp. 105–116.

[64] Holzmüller, D., Zaverkin, V., Kästner, J., and Steinwart, I. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research 24*, 164 (2023), 1–81.

[65] Hosseini Akbarnejad, A. H., Bigras, G., and Ray, N. Gpex, a framework for interpreting artificial neural networks. *Advances in Neural Information Processing Systems 36* (2024).

[66] Hou, L., Samaras, D., Kurc, T. M., Gao, Y., Davis, J. E., and Saltz, J. H. Patch-based convolutional neural network for whole slide tissue image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[67] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[68] Huang, K., Lopez, R., Hutter, J.-C., Kudo, T., Rios, A., and Regev, A. Sequential optimal experimental design of perturbation screens guided by multi-modal priors. *bioRxiv* (2023), 2023–12.

[69] Huang, Y., and Chung, A. C.-s. Improving high resolution histology image classification with deep spatial fusion network. In *Computational Pathology and Ophthalmic Medical Image Analysis: First International Workshop, COMPAY 2018, and 5th International Workshop, OMIA 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings 5* (2018), Springer, pp. 19–26.

[70] Ibrahim, A., Gamble, P., Jaroensri, R., Abdelsamea, M. M., Mermel, C. H., Chen, P.-H. C., and Rakha, E. A. Artificial intelligence in digital breast pathology: techniques and applications. *The Breast 49* (2020), 267–273.

[71] Ilse, M., Tomczak, J., and Welling, M. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning* (10–15 Jul 2018),

J. Dy and A. Krause, Eds., vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 2127–2136.

[72] Jackson, C. R., Sriharan, A., and Vaickus, L. J. A machine learning algorithm for simulating immunohistochemistry: development of sox10 virtual ihc and evaluation on primarily melanocytic neoplasms. *Modern Pathology 33*, 9 (2020), 1638–1648.

[73] Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems 31* (2018).

[74] Jaehoon Lee et al. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems* (2019), vol. 32, Curran Associates, Inc.

[75] Jörke, M., Lee, J., and Brunskill, E. Simple regret minimization for contextual bandits using bayesian optimal experimental design. In *ICML Workshop on "Adaptive Experimental Design and Active Learning in the Real World* (2022), vol. 9.

[76] Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. Active learning with gaussian processes for object categorization. In *2007 IEEE 11th international conference on computer vision* (2007), IEEE, pp. 1–8.

[77] Kather, J. N., Weis, C.-A., Bianconi, F., Melchers, S. M., Schad, L. R., Gaiser, T., Marx, A., and Zöllner, F. G. Multi-class texture analysis in colorectal cancer histology. *Scientific Reports 6* (2016).

[78] Khakzar, A., Khorsandi, P., Nobahari, R., and Navab, N. Do explanations explain? model knows best. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 10244–10253.

[79] Kim, R. H., Nomikou, S., Coudray, N., Jour, G., Dawood, Z., Hong, R., Esteva, E., Sakellaropoulos, T., Donnelly, D., Moran, U., et al. A deep learning approach for rapid mutational screening in melanoma. *BioRxiv* (2019), 610311.

[80] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[81] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. *stat 1050* (2014), 1.

[82] Koh, P. W., and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning* (06–11 Aug 2017), vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 1885–1894.

[83] Krizhevsky, A. Learning multiple layers of features from tiny images. 32–33.

[84] Laleh, N. G., Muti, H. S., Loeffler, C. M. L., Echle, A., Saldanha, O. L., Mahmood, F., Lu, M. Y., Trautwein, C., Langer, R., Dislich, B., et al. Benchmarking weakly-supervised deep learning pipelines for whole slide classification in computational pathology. *Medical image analysis 79* (2022), 102474.

[85] Lin, J. A., Antorán, J., Padhy, S., Janz, D., Hernández-Lobato, J. M., and Terenin, A. Sampling from gaussian process posteriors using stochastic gradient descent. *Advances in Neural Information Processing Systems 36* (2024).

[86] Litjens, G., Bandi, P., Ehteshami Bejnordi, B., Geessink, O., Balkenhol, M., Bult, P., Halilovic, A., Hermsen, M., van de Loo, R., Vogels, R., Manson, Q. F., Stathonikos, N., Baidoshvili, A., van Diest, P., Wauters, C., van Dijk, M., and van der Laak, J. 1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset. *GigaScience 7*, 6 (05 2018), giy065.

[87] Liu, Y., Li, X., Zheng, A., Zhu, X., Liu, S., Hu, M., Luo, Q., Liao, H., Liu, M., He, Y., et al. Predict ki-67 positive cells in h&e-stained images using deep learning independently from ihc-stained images. *Frontiers in Molecular Biosciences 7* (2020), 183.

[88] Loshchilov, I., and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

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

[90] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 4765–4774.

[91] Mamun, O., Taufique, M., Wenzlick, M., Hawk, J., and Devanathan, R. Uncertainty quantification for bayesian active learning in rupture life prediction of ferritic steels. *Scientific Reports 12*, 1 (2022), 2083.

[92] Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations* (2018).

[93] Mills, A. M., Gradecki, S. E., Horton, B. J., Blackwell, R., Moskaluk, C. A., Mandell, J. W., Mills, S. E., and Cathro, H. P. Diagnostic efficiency in digital pathology. *The American journal of surgical pathology 42*, 1 (2018), 53–59.

[94] Mohamadi, M. A., Bae, W., and Sutherland, D. J. Making look-ahead active learning strategies feasible with neural tangent kernels. *Advances in Neural Information Processing Systems 35* (2022), 12542–12553.

[95] Momeni, A., Thibault, M., and Gevaert, O. Deep recurrent attention models for histopathological image analysis. *bioRxiv* (2018).

[96] Mukhopadhyay, S., Feldman, M. D., Abels, E., Ashfaq, R., Beltaifa, S., Cacciabeve, N. G., Cathro, H. P., Cheng, L., Cooper, K., Dickey, G. E., et al. Whole slide imaging versus microscopy for primary diagnosis in surgical pathology: a multicenter blinded randomized noninferiority study of 1992 cases (pivotal study). *The American journal of surgical pathology 42*, 1 (2018), 39.

[97] Neal, R. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer New York, 2012.

[98] Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A., Sohl-dickstein, J., and Schoenholz, S. Neural tangents: Fast and easy infinite neural networks in python.

[99] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics 9*, 1 (1979), 62–66.

[100] Paik, S., Shak, S., Tang, G., Kim, C., Baker, J., Cronin, M., Baehner, F. L., Walker, M. G., Watson, D., Park, T., et al. A multigene assay to predict recurrence of tamoxifen-treated, node-negative breast cancer. *New England Journal of Medicine 351*, 27 (2004), 2817–2826.

[101] Passalis, N., and Tefas, A. Bag-of-features pooling for deep convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2017).

[102] Pop, R., and Fulop, P. Deep ensemble bayesian active learning. *Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada* (2018).

[103] Prewitt, J. M., and Mendelsohn, M. L. The analysis of cell images. *Annals of the New York Academy of Sciences 128*, 3 (1966), 1035–1053.

[104] Qaiser et al., T. Her2 challenge contest: a detailed assessment of automated her2 scoring algorithms in whole slide images of breast cancer tissues. *Histopathology 72*, 2 (2018), 227–238.

[105] Qu, H., Zhou, M., Yan, Z., Wang, H., Rustgi, V. K., Zhang, S., Gevaert, O., and Metaxas, D. N. Genetic mutation and biological pathway prediction based on whole slide images in breast carcinoma using deep learning. *NPJ precision oncology 5*, 1 (2021), 87.

[106] Rainforth, T., Foster, A., Ivanova, D. R., and Bickford Smith, F. Modern bayesian experimental design. *Statistical Science 39*, 1 (2024), 100–114.

[107] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[108] Rasmussen, S. A., Taylor, V. J., Surette, A. P., Barnes, P. J., and Bethune, G. C. Using deep learning to predict final her2 status in invasive breast cancers that are equivocal (2+) by immunohistochemistry. *Applied Immunohistochemistry & Molecular Morphology 30*, 10 (2022), 668–673.

[109] Rawat, R. R., Ortega, I., Roy, P., Sha, F., Shibata, D., Ruderman, D., and Agus, D. B. Deep learned tissue "fingerprints" classify breast cancers by er/pr/her2 status from h&e images. *Scientific reports 10*, 1 (2020), 7275.

[110] Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations* (2018).

[111] A. Wilson et al. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems* (2016), vol. 29, Curran Associates, Inc.

[112] D. Slack et al. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (New York, NY, USA, 2020), AIES '20, Association for Computing Machinery, p. 180–186.

[113] J. Gardner et al. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems 2018-December* (2018), 7576–7586.

[114] V. Dutordoir et al. Deep neural networks as point estimates for deep gaussian processes. *Advances in Neural Information Processing Systems 34* (2021), 9443–9455.

[115] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016* (2016), pp. 1135–1144.

[116] Riquelme, C. *Online active learning with linear models*. PhD thesis, Stanford University, 2017.

[117] Ruifrok, A. C., Johnston, D. A., et al. Quantification of histochemical staining by color deconvolution. *Analytical and quantitative cytology and histology 23*, 4 (2001), 291–299.

[118] Schmidt, U., Weigert, M., Broaddus, C., and Myers, G. Cell detection with star-convex polygons. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II* (2018), pp. 265–273.

[119] Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H., and Toussaint, M. Safe exploration for active learning with gaussian processes. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III 15* (2015), Springer, pp. 133–149.

[120] Settles, B. Active learning literature survey.

[121] Shaban, M., Awan, R., Fraz, M. M., Azam, A., Tsang, Y.-W., Snead, D., and Rajpoot, N. M. Context-aware convolutional neural network for grading of colorectal cancer histology images. *IEEE transactions on medical imaging 39*, 7 (2020), 2395–2405.

[122] Shamai, G., Binenbaum, Y., Slossberg, R., Duek, I., Gil, Z., and Kimmel, R. Artificial intelligence algorithms to assess hormonal status from tissue microarrays in patients with breast cancer. *JAMA network open 2*, 7 (2019), e197700–e197700.

[123] Shamai, G., Livne, A., Polónia, A., Sabo, E., Cretu, A., Bar-Sela, G., and Kimmel, R. Deep learning-based image analysis predicts pd-l1 status from h&e-stained histopathology images in breast cancer. *Nature Communications 13*, 1 (2022), 6753.

[124] Shamai, G., Schley, R., Kimmel, R., Balint-Lahat, N., Barshack, I., and Mayer, C. Abstract 5354: Prediction of OncotypeDX high risk group for chemotherapy benefit in breast cancer by deep learning analysis of hematoxylin and eosin-stained whole slide images. *Cancer Research 83*, 7-Supplement (04 2023), 5354–5354.

[125] Sharma, Y., Shrivastava, A., Ehsan, L., Moskaluk, C. A., Syed, S., and Brown, D. Cluster-to-conquer: A framework for end-to-end multi-instance learning for whole slide image classification. In *Medical Imaging with Deep Learning* (2021), PMLR, pp. 682–698.

[126] Smith, F. B., Kirsch, A., Farquhar, S., Gal, Y., Foster, A., and Rainforth, T. Prediction-oriented bayesian active learning. In *International Conference on Artificial Intelligence and Statistics* (2023), PMLR, pp. 7331–7348.

[127] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[128] Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning* (06–11 Aug 2017), vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 3319–3328.

[129] Tavolara, T. E., Niazi, M., Gower, A. C., Ginese, M., Beamer, G., and Gurcan, M. N. Deep learning predicts gene expression as an intermediate data modality to identify susceptibility patterns in mycobacterium tuberculosis infected diversity outbred mice. *EBioMedicine 67* (2021), 103388.

[130] Tellez, D., Höppener, D., Verhoef, C., Grünhagen, D., Nierop, P., Drozdzal, M., van der Laak, J., and Ciompi, F. Extending unsupervised neural image compression with supervised multitask learning. In *Proceedings of the Third Conference on Medical Imaging with Deep Learning* (06–08 Jul 2020), T. Arbel, I. Ben Ayed, M. de Bruijne, M. Descoteaux, H. Lombaert, and C. Pal, Eds., vol. 121 of *Proceedings of Machine Learning Research*, PMLR, pp. 770–783.

[131] Vutukuri, H. Dogs vs Wolves Classification of Dogs and Wolves. https://www.kaggle.com/harishvutukuri/dogs-vs-wolves, 2019. [Online; accessed 19-Dec-2021].

[132] Wilson, A., and Nickisch, H. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning* (2015), PMLR, pp. 1775–1784.

[133] Y. Bengio's post on gp vs ann https://qr.ae/pvqZn7.

[134] Yan, R., Ren, F., Wang, Z., Wang, L., Zhang, T., Liu, Y., Rao, X., Zheng, C., and Zhang, F. Breast cancer histopathological image classification using a hybrid deep neural network. *Methods 173* (2020), 52–60.

[135] Yang, Y., Akbarnejad, A., Ray, N., and Bigras, G. Double adversarial domain adaptation for whole-slide-imageclassification. In *Medical Imaging with Deep Learning* (2021).

[136] Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems 31* (2018).

[137] Zeng, B., Lin, Y., Wang, Y., Chen, Y., Dong, J., Li, X., and Zhang, Y. Semi-supervised pr virtual staining for breast histopathological images. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part II* (2022), Springer, pp. 232–241.

[138] Zhang, C., Song, Y., Zhang, D., Liu, S., Chen, M., and Cai, W. Whole slide image classification via iterative patch labelling. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), pp. 1408–1412.

[139] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2921–2929.

[140] Zhou, Y., Graham, S., Alemi Koohbanani, N., Shaban, M., Heng, P.-A., and Rajpoot, N. Cgc-net: Cell graph convolutional network for grading of colorectal cancer histology images. In *Proceedings of the IEEE/CVF international conference on computer vision workshops* (2019), pp. 0–0.

[141] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2223–2232.

Appendices

Appendix A

The Proposed Method of Sec. 4.3 Applied for Recurrence Score Prediction

In this appendix we evaluate the proposed method of Sec. 4.3 in predicting breast cancer recurrence score from H&E whole-slide images. The experiments of this section were done by Ms. Namitha Guruprasad (the first author of [57]).

At the time our private recurrence-score dataset contained 558 H&-stained whole-slide images from 558 patients, and for each patient OncotypeDX® recurrence-score (a number between 0 and 100) was available. We considered the WSI classification problem 4.2 where patients with recurrence-score below (resp. above) 25.5 were labeled as class 0 (resp. 1). We considered three random splits to train/testing sets, 75% of patients were included in the training/validation set and the rest were included in the testing set. To inspect how adding more and more training instances can improve the prediction performance, we started with an initial training set containing only 10% of patients, and repeatedly enlarged the training set by including 5% more patients until the training set contains 75% of patients in the whole dataset. This gradual increase in the size of training set is shown in the horizontal axes of Figs. A.1a, A.1b, A.1c, and A.1d. Three methods were evaluated in this setting:

- The state-of-the-art method CLAM [89], which is labeled as "CLAM" in Fig. A.1.
- The proposed method of Sec. 4.3 with the parameter settings described in Sec. 4.4, but with more training iterations (400K). This method is labeled as "Proposed" in Fig. A.1.
- 13000 patches - each of size 1500 by 1500 pixels - were extracted from WSIs in the recurrence-score dataset. Afterwards, the cell nuclei in each patch were found by the segmentation

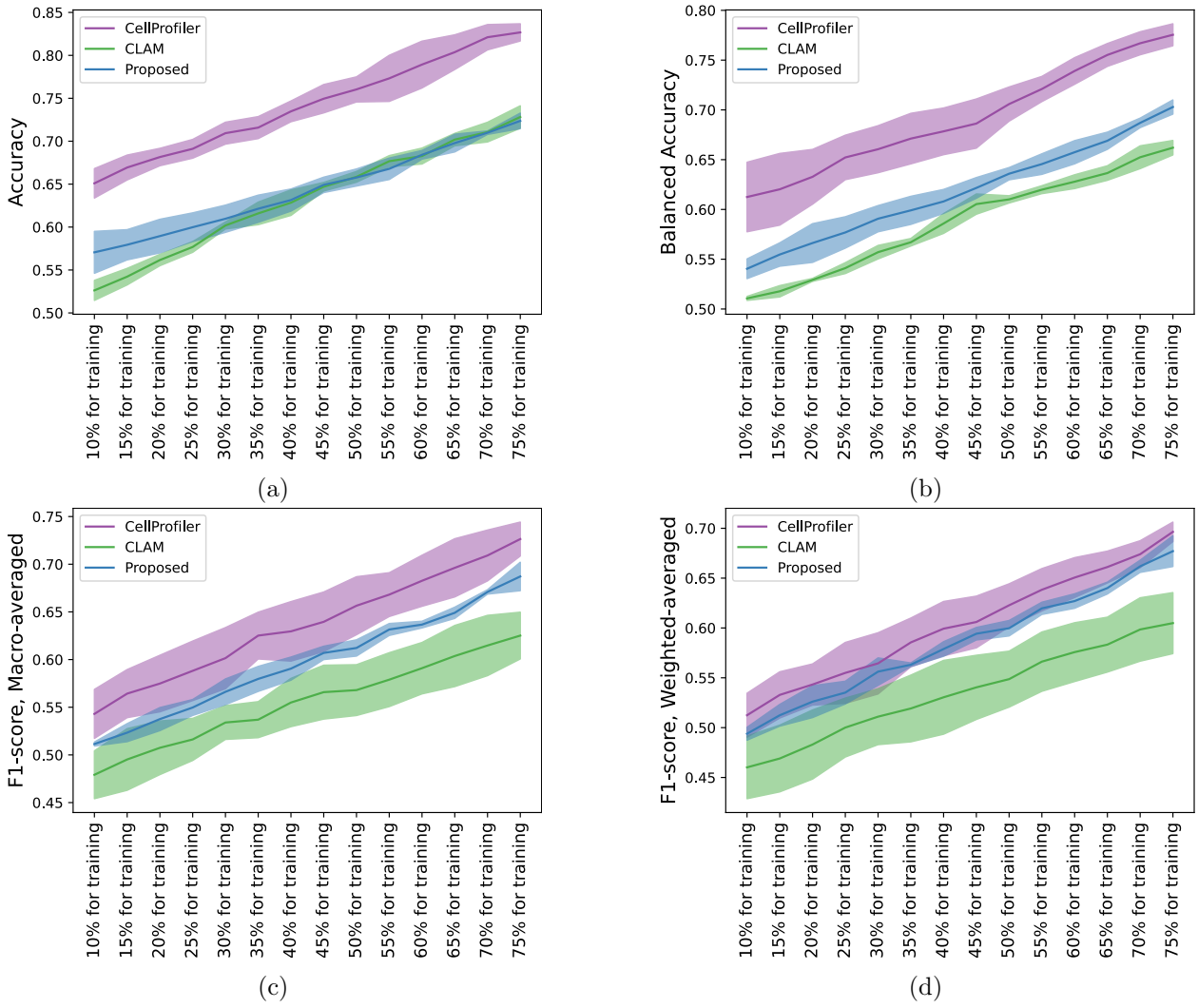


Figure A.1: Three methods evaluated in predicting recurrence-score from H&E-stained whole-slide images. Prediction performance on the testing set is measured by a) accuracy, b) balanced accuracy, c) F1-score, macro-averaged, and d) F1-score, weighted-averaged. As seen in the horizontal axes, initially 10% of patients in the dataset are selected as the training set. Consequently 5% of the dataset is repeatedly added to the training set to see how the prediction performance is increased.

method StarDist [118]. After obtaining more than a million nuclei, some commonly used nuclei features in CellProfiler [40] were implemented in OpenCV and were extracted for each nuclei. For each nucleus the following shape features were extracted: area, perimeter, form factor, solidity, eccentricity, and Zerenik feature. Moreover, for each nucleus the following texture features were extracted both for unwrapped and radially warped nucleus image: total intensity, mean intensity, variation, and intensity moments. Afterwards the multiple-instance learning package mil [13] was adopted for WSI classification. In Fig. A.1 this method is labeled as "CellProfiler".

In Fig. A.1 we observed that almost in all settings the proposed method of Sec. 4.3 outperforms the state-of-the-art method CLAM [89]. Moreover, interestingly in Fig. A.1 we observe that the hand-engineered CellProfiler [40] features outperform both CLAM [89] and the proposed method of Sec. 4.3.

Appendix B

Deriving the Variational Lower-bound for GPEX

In this section we derive the variational lower-bound introduced in Sec. 6.3.3. We firstly introduce Lemmas 1 and 2 as they appear in our derivations.

Lemma 1. *The KL-divergence between two normal distributions* $\mathcal{N}_1(.\ ;\ \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\mathcal{N}_2(.\ ;\ \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *can be computed as follows:*

$$KL\Big(\mathcal{N}_1 \,||\, \mathcal{N}_2\Big) = \frac{1}{2}\Big(\log(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}) \ - \ D \ + \ trace\{\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\} + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\Big).\blacksquare \tag{B.1}$$

Lemma 2. *Let* p_1 *and* p_2 *be two normal distributions:*

$$p_1(x) = \mathcal{N}\big(x\ ;\ \mu_1, \sigma_1^2\big),$$
$$p_2(x) = \mathcal{N}\big(x\ ;\ \mu_2, \sigma_2^2\big).$$

We have that

$$\mathbb{E}_{x\sim p_2}\big[\log\ p_1(x\ ;\ \mu_1, \sigma_1^2)\big] = -\frac{(\mu_1-\mu_2)^2+\sigma_2^2}{2\sigma_1^2} - \frac{1}{2}\log(\sigma_1^2) - \frac{1}{2}\log(2\pi).\blacksquare \tag{B.2}$$

Fig. B.1 illustrates the framework as a probabilistic graphical model. A general feed-forward pipeline takes in a set of input(s) $\mathcal{X}$ and produces a set of output(s) $\mathcal{Y}$. The general pipeline is required to have at least one ANN as a submodule. The ANN submodule is required to take in only one input $\boldsymbol{x}$ and to produce only one output $\boldsymbol{v}$, where $\boldsymbol{x}$ and $\boldsymbol{v}$ are tensors of arbitrary sizes.

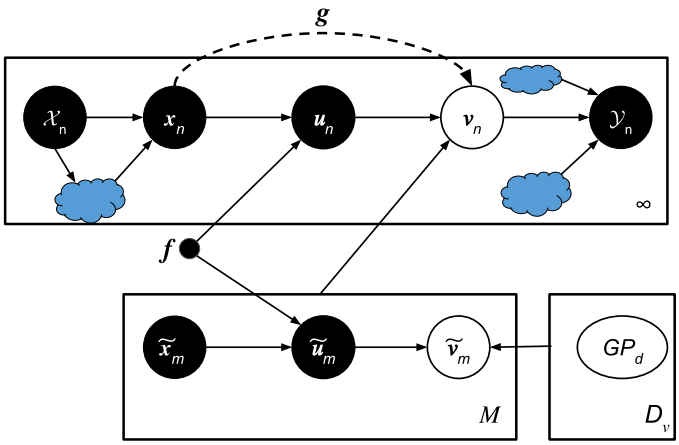


Figure B.1: The proposed framework as a probabilistic graphical model.

As illustrated in Fig. B.1, the ANN's input $\boldsymbol{x}$ can depend arbitrarily on some other intermediate variables in the pipeline. This relation is modeled by the conditional distribution $p\big(\boldsymbol{x}_n|Parent(\boldsymbol{x}_n)\big)$ where $Parent(\boldsymbol{x}_n)$ is the set of all variables which are connected to $\boldsymbol{x}_n$. Similarly, as illustrated in Fig. B.1 the pipeline's output $\mathcal{Y}$ can arbitrarily depend on some intermediate variables in the pipeline. This relation is modeled by the conditional distribution $p\big(\mathcal{Y}_n|Parent(\mathcal{Y}_n)\big)$. In Fig. B.1 the lower boxes are the inducing points and other variables that determine the GPs' posterior. More precisely, in Fig. B.1 the variables $\{\tilde{\boldsymbol{x}}_m\}_{m=1}^M$ are some inducing points (e.g. some training images). Vectors in the kernel space are denoted by $\tilde{\boldsymbol{u}}$ and $\boldsymbol{u}$. Moreover, the observed values are denoted by v and $\tilde{v}$. Informally, $\boldsymbol{u}$ and v denote the input/output of the GPs. When referring to one of the M inducing points a "tilde" is used (as $(\tilde{\boldsymbol{u}}, \tilde{v})$), however $(\boldsymbol{u}, v)$ corresponds to a point that can be anywhere in the kernel-space.

The inducing instances $\{\tilde{\boldsymbol{x}}_m\}_{m=1}^M$ are mapped to the kernel-spaces by the kernel mappings $\{f_1(.), ..., f_L(.)\}$. In Fig.B.1 the variables $\{\tilde{\boldsymbol{u}}_m\}_{m=1}^M$ are the kernel-space representations of the inducing points $\{\tilde{\boldsymbol{x}}_m\}_{m=1}^M$. Moreover, $\{\tilde{\boldsymbol{v}}_m\}_{m=1}^M$ are the GP's output values at the inducing points. Given an instance $\boldsymbol{x}_n$, it is firstly fed to the kernel mappings $\{f_1(.), ..., f_L(.)\}$ and the kernel-space representations $\boldsymbol{u}_n$ are obtained. Afterwards, the GPs' outputs on $\boldsymbol{u}_n$ depend on $\boldsymbol{u}_n$ as well as all other inducing points because the inducing points actually determine the GPs' posterior on all kernel-space points including $\boldsymbol{u}_n$. Therefore, in Fig. B.1 the variable $\boldsymbol{v}_n$ is not only connected to $\boldsymbol{u}_n$ but it is also connected to the box at the bottom (*i.e.* all inducing points and other variables associated with them).

As usual, the variational lower-bound is equal to

$$\mathcal{L} = \mathbb{E}_{\sim q}\big[\log p(\text{all variables})\big] \; - \; \mathbb{E}_{\sim q}\big[\log q(\text{hidden variables})\big]. \tag{B.3}$$

The likelihood of all variables in Eq. B.3 factorizes as the product of conditional distributions of each variable given its parents. Therefore

$$p(\text{all variables}) = \prod_{variable\ t} p\big(t|Parent(t)\big). \tag{B.4}$$

In Eq. B.4 only some conditional distributions appear in our derivations which are discussed at the following.

- The variable $\boldsymbol{x}_n$: the ANN's input $\boldsymbol{x}_n$ can depend arbitrarily on some other intermediate variables in the pipeline. In our derivations we leave this conditional distribution as $p\big(\boldsymbol{x}_n|Parent(\boldsymbol{x}_n)\big)$.

- The variable $\boldsymbol{u}_n$: Given a training instance $\boldsymbol{x}_n$, the kernel-space representations $\boldsymbol{u}_n$ are deterministically obtained by feeding the instance to the kernel-mappings $[f_1(.), ..., f_L(.)]$.

- The variable $\boldsymbol{v}_n$: The ANN's output is required to depend only on the input, so

$$p\big(\boldsymbol{v}_n|Parent(\boldsymbol{v}_n)\big) = p\big(\boldsymbol{v}_n|\boldsymbol{u}_n, \boldsymbol{x}_n, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^M\big). \tag{B.5}$$

  The above distribution is actually the GPs' posterior at $\boldsymbol{u}_n$ (i.e. the normal distribution of Eq. 6.1).

- The variable $\tilde{\boldsymbol{x}}_m$: the inducing point $\tilde{\boldsymbol{x}}_m$ can depend arbitrarily on some other intermediate variables in the pipeline. In our derivations we leave this conditional distribution as $p\big(\tilde{\boldsymbol{x}}_m|Parent(\tilde{\boldsymbol{x}}_m)\big)$.

- The variable $\tilde{\boldsymbol{u}}_m$: Given an inducing point $\tilde{\boldsymbol{x}}_m$, the kernel-space representations $\tilde{\boldsymbol{u}}_m$ are deterministically obtained by feeding the inducing point $\tilde{\boldsymbol{x}}_m$ to the kernel-mappings $[f_1(.), ..., f_L(.)]$.

- The variables $\hat{\boldsymbol{v}}_m$: Given the kernel-space representations $\{\tilde{\boldsymbol{u}}_m^{(\ell)}\}_{m=1}^M$, the variables $\{\tilde{v}_1^{(\ell)}, ..., \tilde{v}_M^{(\ell)}\}$ follow a M-dimensional Gaussian distribution with zero mean and a covariance matrix determined by the GP prior covariance among the variables $\{\tilde{\boldsymbol{u}}_m^{(\ell)}\}_{m=1}^M$.

- The variable $\mathcal{Y}_n$: the pipeline's output $\mathcal{Y}$ can arbitrarily depend on some intermediate variables in the pipeline. In our derivations we leave this conditional distribution as $p\big(\mathcal{Y}_n|Parent(\mathcal{Y}_n)\big)$.

According to Eq. B.4, the likelihood of all variables factorizes as

$$
\begin{aligned}
p(\text{all variables}) &= \prod_{\text{variable } t} p\big(t|Parent(t)\big) \\
&= \big(\prod_{n} p(\boldsymbol{x}_n|Parent(\boldsymbol{x}_n))\big) \times \big(\prod_{n} p(\boldsymbol{u}_n|\boldsymbol{x}_n)\big)\times \\
&\big(\prod_{n}\prod_{\ell} p(v_n^{(\ell)}|\boldsymbol{u}_n, \boldsymbol{x}_n, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M})\big)\times \\
&\big(\prod_{m} p(\tilde{\boldsymbol{x}}_m|Parent(\tilde{\boldsymbol{x}}_m))\big) \times \big(\prod_{m} p(\tilde{\boldsymbol{u}}_m|\tilde{\boldsymbol{x}}_m)\big)\times \\
&\big(\prod_{\ell} p(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)}|\boldsymbol{0}, \mathcal{K}_{prior}(\tilde{\boldsymbol{u}}_{1:M}^{(\ell)}, \tilde{\boldsymbol{u}}_{1:M}^{(\ell)}))\big)\times \\
&\big(\prod_{n} p(\mathcal{Y}_n|Parent(\mathcal{Y}_n))\big) \times \big(\prod_{\text{other vars } t} p\big(t|Parent(t)\big)\big).
\end{aligned}
\tag{B.6}
$$

Now we derive the lower-bound $\mathcal{L}$ with respect to each parameter separately.

B.0.1 Deriving the Lower-bound With Respect to the Kernel-mappings

In the right-hand-side of Eq. B.6 only the following terms are dependant on the kernel-mappings $[f_1(.), ..., f_L(.)]$:

$$
\begin{aligned}
&\big[\prod_{m} p(\tilde{\boldsymbol{u}}_m|\tilde{\boldsymbol{x}}_m) \times \prod_{\ell} p(\tilde{v}_m^{(\ell)}|\boldsymbol{0}, \mathcal{K}_{prior}(\tilde{\boldsymbol{u}}_{1:M}^{(\ell)}, \tilde{\boldsymbol{u}}_{1:M}^{(\ell)}))\big]\times \\
&\big[\prod_{n} p\big(\boldsymbol{u}_n|\boldsymbol{x}_n\big) \times \prod_{\ell} p\big(v_n^{(\ell)}|\boldsymbol{u}_n, \boldsymbol{x}_n, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M}\big)\big].
\end{aligned}
\tag{B.7}
$$

Note that in the above equation the terms $p(\tilde{\boldsymbol{u}}_m|\tilde{\boldsymbol{x}}_m)$ and $p(\boldsymbol{u}_n|\boldsymbol{x}_n)$ are equal to 1 because $\tilde{\boldsymbol{u}}_m$ and $\boldsymbol{u}_n$ are deterministically obtained from $\tilde{\boldsymbol{x}}_m$ and $\boldsymbol{x}_n$. Therefore, in Eq. B.3 the terms containing the kernel mappings $[f_1(.), ..., f_L(.)]$ are as follows:

$$
\begin{aligned}
\mathcal{L}_f &= \mathbb{E}_{\sim q}\big[\sum_{\ell} \log p(v^{(\ell)}|\boldsymbol{u}, \boldsymbol{x}, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M})\big] + \\
&\quad \sum_{\ell} \mathbb{E}_{\sim q}\big[\log p(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)}|\boldsymbol{0}, \mathcal{K}_{prior}(\tilde{\boldsymbol{u}}_{1:M}^{(\ell)}, \tilde{\boldsymbol{u}}_{1:M}^{(\ell)}))\big] - \\
&\quad \sum_{\ell} \mathbb{E}_{\sim q}\big[\log q_2(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)})\big] \\
&= \mathbb{E}_{\sim q}\big[\sum_{\ell} \log p(v^{(\ell)}|\boldsymbol{u}, \boldsymbol{x}, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M})\big] - \\
&\sum_{\ell=1} \mathbb{E}_{\sim q}\big[KL\Big(q_2(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)}) \;\; || \;\; p(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)}|\boldsymbol{0}, \mathcal{K}_{prior}(\tilde{\boldsymbol{u}}_{1:M}^{(\ell)}, \tilde{\boldsymbol{u}}_{1:M}^{(\ell)}))\Big)\big].
\end{aligned}
\tag{B.8}
$$

We simplify the two terms on the right-hand-side of Eq. B.8. The first term is the expected log-likelihood of a Gaussian distribution (i.e. the conditional log-likelihood of $\tilde{v}^{\ell}$ as in Eq. 6.1). Also the variational distribution $q(.)$ is Gaussian. Therefore, we can use Lemma.2 to simplify the first term:

$$
\begin{aligned}
&\mathbb{E}_{\sim q}\big[\sum_{\ell=1}^{L}\log p(v^{(\ell)}|\boldsymbol{u},\boldsymbol{x},\{\tilde{\boldsymbol{x}}_m,\tilde{\boldsymbol{u}}_m,\tilde{v}_m\}_{m=1}^{M})\big] = \\
&\sum_{\ell=1}^{L}\mathbb{E}_{\sim q}\big[\log p(v^{(\ell)}|\boldsymbol{u},\boldsymbol{x},\{\tilde{\boldsymbol{x}}_m,\tilde{\boldsymbol{u}}_m,\tilde{v}_m\}_{m=1}^{M})\big] = \\
&\sum_{\ell=1}^{L}\Big[-\frac{\big(\mu_v(\boldsymbol{u}^{(\ell)},\tilde{\mathbf{u}}_{1:M}^{(\ell)},\tilde{v}_{1:M}^{(\ell)})-g_\ell(\boldsymbol{x})\big)^2+\sigma_g^2}{cov_v(\boldsymbol{u}^{(\ell)},\tilde{\mathbf{u}}_{1:M}^{(\ell)},\tilde{v}_{1:M}^{(\ell)})} \\
&\qquad-\frac{1}{2}\log\big(cov_v(\boldsymbol{u}^{(\ell)},\tilde{\mathbf{u}}_{1:M}^{(\ell)},\tilde{v}_{1:M}^{(\ell)})\big) \\
&\qquad-\frac{1}{2}\log(2\pi)\Big].
\end{aligned}
\tag{B.9}
$$

Note that the two terms of Eq. B.9 are the two terms which were presented and discussed in Eq. 6.5.

Now we simplify the KL-term on the right-hand-side of Eq. B.8. According to Lemma.1 we have that

$$
\begin{aligned}
KL\Big(q_2(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)})\ ||\ p(\tilde{\boldsymbol{v}}_{1:M}^{(\ell)}|\mathbf{0},\mathcal{K}_{prior}(\tilde{\boldsymbol{u}}_{1:M}^{(\ell)},\tilde{\boldsymbol{u}}_{1:M}^{(\ell)}))\Big) = \\
&+0.5\big(\log(\frac{\sigma_{gp}^2}{\sigma_\varphi^2})\big) \\
&-0.5M \\
&+\frac{\sigma_\varphi^2}{\sigma_{gp}^2} \\
&+\frac{\boldsymbol{\varphi}_{1:M}^{(\ell)^T}\boldsymbol{\varphi}_{1:M}^{(\ell)}}{\sigma_{gp}^2},
\end{aligned}
\tag{B.10}
$$

where φ are the variational parameters of $q_2(.)$ as in Eq. 6.4. Therefore, the KL-term of Eq. B.8 is a constant with respect to the kernel mappings $[f_1(.),...,f_L(.)]$ and can be discarded. All in all, the lower-bound for optimizing the kernel-mappings is equal to the right-hand-side of Eq. B.9 which was introduced and discussed in Sec. 6.3.3.

B.0.2 Deriving the Lower-bound With Respect to the ANN Parameters

According to Eq. 6.4 , in our formulation the ANN's parameters appear as some variational parameters. Therefore, the likelihood of all variables (Eq. B.6) does not generally depend on the ANN's parameters. But according to the general ELBO formulation in Eq. B.3 the ELBO $\mathcal{L}$ depends on ANN's parameters, because when computing the expectation the variables are drawn from the variational distribution $q(.)$. We estimated the ELBO of Eq. B.3 by the average over few samples. More precisely, given a training instance $\boldsymbol{x}$, we firstly computed the kernel-space representations as:

$$\boldsymbol{u}^{(\ell)} = f_\ell(\boldsymbol{x}), \;\; 1 \leq \ell \leq L. \tag{B.11}$$

Afterwards, we used the reparametrization trick for Eq. 6.1 to draw a sample for $\boldsymbol{v}^{(\ell)}$ as follows:

$$\begin{aligned} z_{q2}^{(\ell)} &\sim \mathcal{N}(0,1), \\ v^{(\ell)} &\sim \mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}) + z_{q2}^{(\ell)} cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}), \end{aligned} \tag{B.12}$$

where $\mu_v(.,.,.)$ and $cov_v(.,.,.)$ are defined in Eqs. 6.2 and 6.3. Moreover, we continue the forward pass of the original pipeline to get a sample $\mathcal{Y}$. Having drawn $\boldsymbol{x}$, $\boldsymbol{u}$, v, and $\mathcal{Y}$ from the variational distribution, we estimate the ELBO of Eq. B.3 by these samples.

$$\begin{aligned} &\mathcal{L} = \\ &\mathbb{E}_{\sim q}\big[\log p(\text{all variables})\big] - \mathbb{E}_{\sim q}\big[\log q(\text{hidden variables})\big] \\ &\approx \log p(\text{all variables})\Big|_{\boldsymbol{x},\boldsymbol{u},v,\mathcal{Y}} - \sum_{m}^{M}\sum_{\ell}^{L} \mathbb{E}_{\sim q_2}\big[\log q_2(\tilde{v}_m^{(\ell)})\big] \end{aligned} \tag{B.13}$$

In the above equation, the second term on the right-hand-side is the entropy of a normal distribution and it only depends on the variance of the q_2 distribution. As we let the variance of q_2 be fixed (σ_g^2 in Eq. 6.4), the second term is a constant. Therefore,

$$\mathcal{L} \approx \log p(\text{all variables})\Big|_{\boldsymbol{x},\boldsymbol{u},v,\mathcal{Y}}. \tag{B.14}$$

Among the likelihood term on the right-hand-side of Eq. B.6 the conditional distribution of all variables before $\boldsymbol{u}_n$ (e.g. $\boldsymbol{x}_n$ and $\mathcal{X}_n$) are independent of the ANN's parameters (i.e. the parameters of the function $g(.)$). On the other hand, for all variables that appear after $\boldsymbol{u}_n$, the conditional

distribution depends on the ANN's parameters. Indeed, according to Eq. B.14

$$\begin{aligned}\mathcal{L}_{ann} \approx & \big[\sum_{\ell=1}^{L} \log p(v^{(\ell)}|\boldsymbol{u}, \boldsymbol{x}, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M})\big]\Big|_{\boldsymbol{x},\boldsymbol{v}} + \\ & \log p(\mathcal{Y}|Parent(\mathcal{Y}))\Big|_{\boldsymbol{x},\boldsymbol{v},\mathcal{Y}} + \\ & \big(\sum_{\text{other vars after } \boldsymbol{u}_n} \log p(t|Parent(t))\big)\Big|_{\boldsymbol{x},\boldsymbol{v},\mathcal{Y}}\big).\end{aligned} \tag{B.15}$$

In the above equation, the first term on the right-hand-side is the log-likelihood of the normal distribution of Eq. 6.1:

$$\begin{aligned}&\log p(v^{(\ell)}|\boldsymbol{u}, \boldsymbol{x}, \{\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{u}}_m, \tilde{v}_m\}_{m=1}^{M}) = \\ &-\frac{1}{2}\big[\sum_{\ell=1}^{L} \frac{(\mu_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)}) - g_\ell(\boldsymbol{x}))^2}{cov_v(\boldsymbol{u}^{(\ell)}, \tilde{\mathbf{u}}_{1:M}^{(\ell)}, \tilde{v}_{1:M}^{(\ell)})}\big] \\ &+ \big(\text{some terms independent from } g(.)\big).\end{aligned} \tag{B.16}$$

In Eq. B.15 the term $p\big(\mathcal{Y}|Parent(\mathcal{Y})\big)$ is the likelihood of the output(s) of the whole pipeline as illustrated by Fig. 6.1a, given the ANN's output and all other intermediate variables on which the final output $\mathcal{Y}$ depends. This likelihood turns out to be equivalent to commonly-used losses like the cross-entropy loss or the mean-squared loss. Here we elaborate upon how this happens. Let the task be a classification, and let $\hat{\mathcal{Y}} \in \mathbb{R}^L$ be the pipeline's output. The final model prediction $\mathcal{Y}$ is done as follows:

$$\mathcal{Y} \sim \mathcal{C}ategorical(\hat{\mathcal{Y}}_K, ..., \hat{\mathcal{Y}}_K) \tag{B.17}$$

Therefore we have that

$$p\big(\mathcal{Y}|Parent(\mathcal{Y})\big) = (\hat{\mathcal{Y}}_1)^{I[\mathcal{Y}==1]} \times ... \times (\hat{\mathcal{Y}}_K)^{I[\mathcal{Y}==K]}, \tag{B.18}$$

where $I[.]$ is the indicator function. So, we have that

$$\begin{aligned}&\log p\big(\mathcal{Y}|Parent(\mathcal{Y})\big) = \\ &I[\mathcal{Y} == 1]\log(\hat{\mathcal{Y}}_1) + ... + I[\mathcal{Y} == K]\log(\hat{\mathcal{Y}}_K).\end{aligned} \tag{B.19}$$

Therefore, when the pipeline is for classification, $\log p(\mathcal{Y}|\mathbf{v},\ etc.)$ will be equal to the cross-entropy loss. This conclusion was introduced and discussed under Eq. 6.6. We can draw similar conclusions when the pipeline is for other tasks like regression, or even a combination of tasks.

In the general pipeline of Fig. B.1 if all stages after $\boldsymbol{v}$ are deterministic (of course except the final stage which is probabilistic like Eq. B.17), the third term on the right-hand-side of Eq. B.15

becomes 1. Therefore, the right-hand-side of Eq. B.15 is equal to Eq. 6.6. As we discussed in Sec. 6.3.3, $\mathcal{L}_{ann}$ has two terms: the first terms encourages the GP-ANN analogy and the second term seeks to lower the task-loss.

B.0.3 Deriving the Lower-bound With Respect to $q_2(.)$ Parameters

In Eq. 6.4 we considered the variational parameters $\{\varphi_m^{(\ell)}\}_{m=1}^M$ for the hidden variables $\{\tilde{v}_m^{(\ell)}\}_{m=1}^M$. The ELBO of Eq. B.3 can be optimized with respect to $\{\varphi_m^{(\ell)}\}_{m=1}^M$ as well. But we noticed that optimizing $\{\varphi_m^{(\ell)}\}_{m=1}^M$ is computationally unstable. Therefore, we set $\{\varphi_m^{(\ell)}\}_{m=1}^M$ according to the following rule:

$$\begin{aligned} &\varphi_m^{(\ell)} = g_\ell(\tilde{\boldsymbol{x}}_m), \\ &\quad 1 \leq m \leq M, \;\; 1 \leq \ell \leq L. \end{aligned} \tag{B.20}$$

We set $\{\varphi_m^{(\ell)}\}_{m=1}^M$ as above because $\tilde{v}_m^{(\ell)}$ is simply the ℓ-th GP posterior mean at the inducing point $\tilde{\boldsymbol{x}}_m$. In other words, to make the GP's posterior mean equal to the ANN's output, $\tilde{v}_\ell^{(m)}$ should be equal to the ANN's (i.e. $g(.)$'s) output at the m-th inducing point.

Appendix C

GPEX Results, Additional Figures and Plots

C.1 Measuring Faithfulness of GPs to ANNs

This section contains additional figures for Sec. 6.4.1.

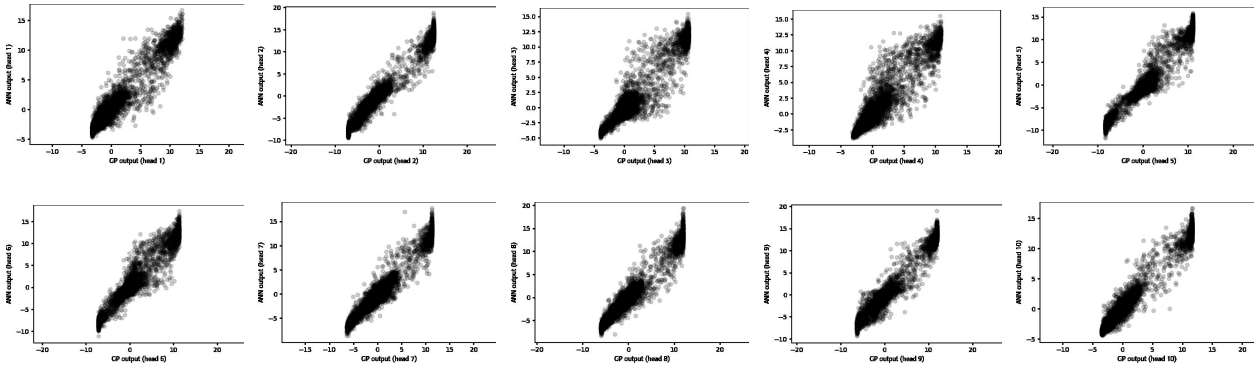


Figure C.1: Scatters for Cifar10 (classifier).

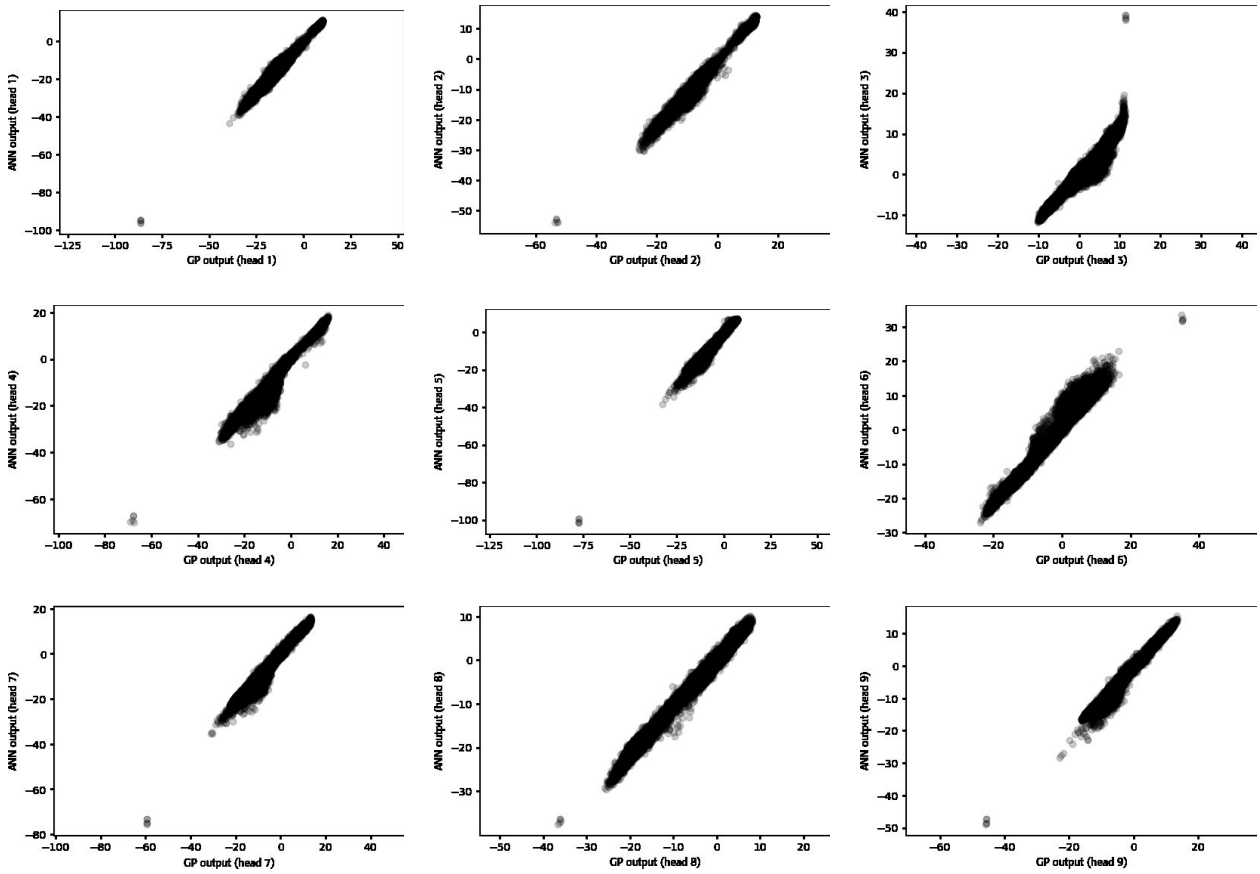


Figure C.2: Scatters for Kather dataset (classifier).

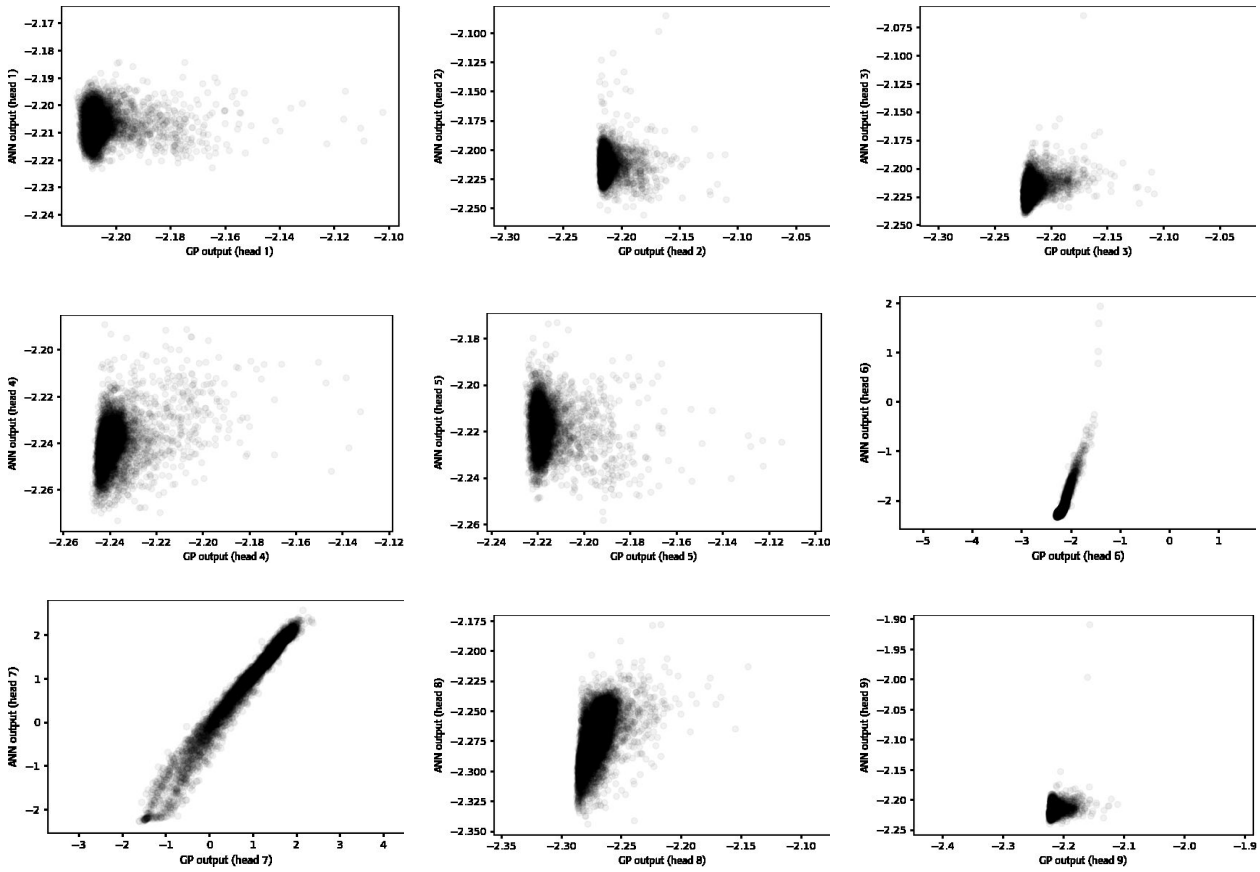


Figure C.3: Scatters for Cifar10 (attention).

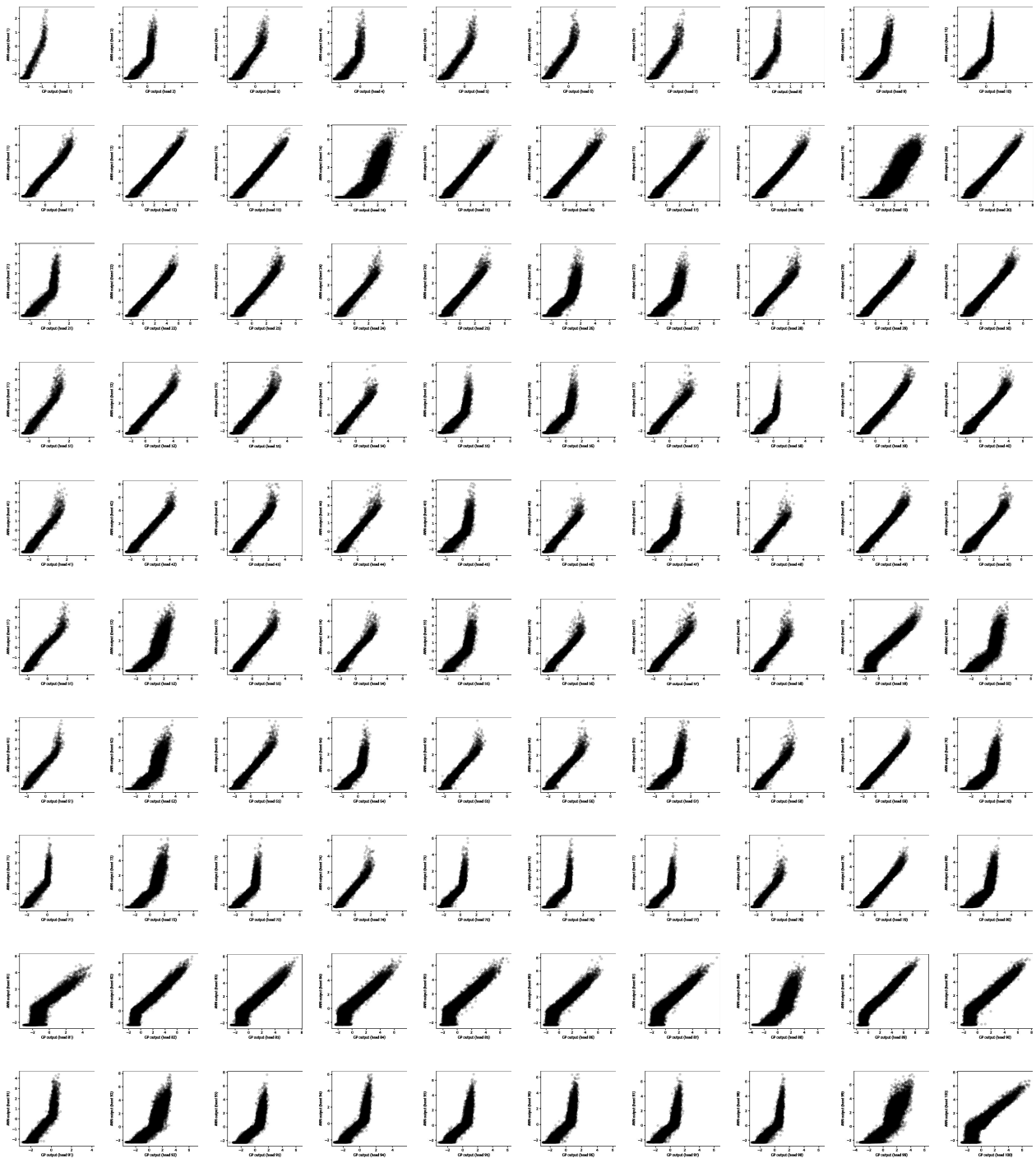


Figure C.4: Scatters for Kather dataset (attention).

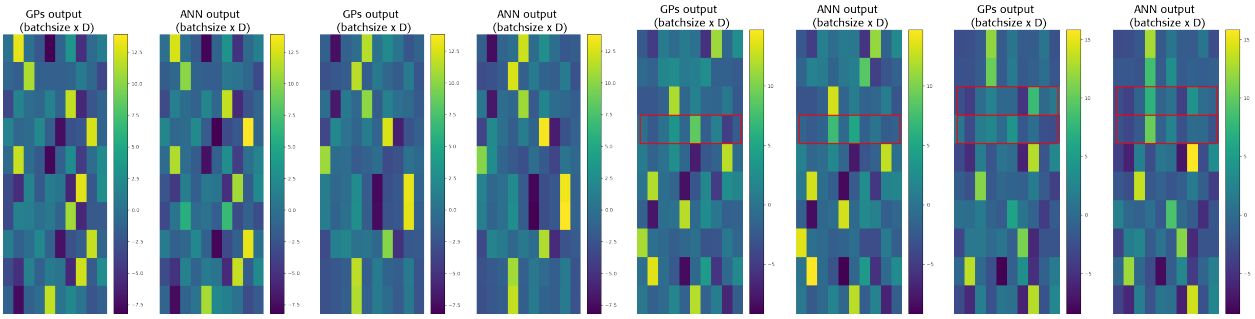


Figure C.5: Comparing GP and ANN outputs for four minibatches of Cifar10 [83] dataset. The red rectangles highlight the instnaces for which the predictions of GP and ANN (i.e. the class with maximum score) are different.

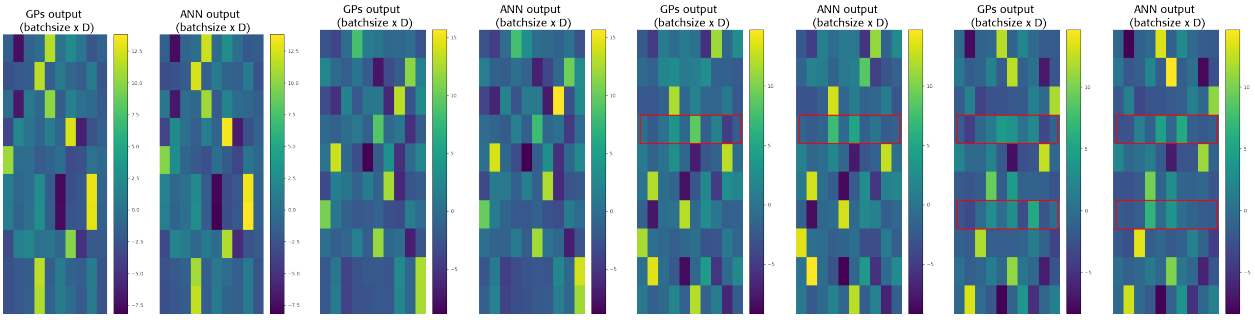


Figure C.6: Comparing GP and ANN outputs for four minibatches of Kather [77] dataset. The red rectangles highlight the instnaces for which the predictions of GP and ANN (i.e. the class with maximum score) are different.

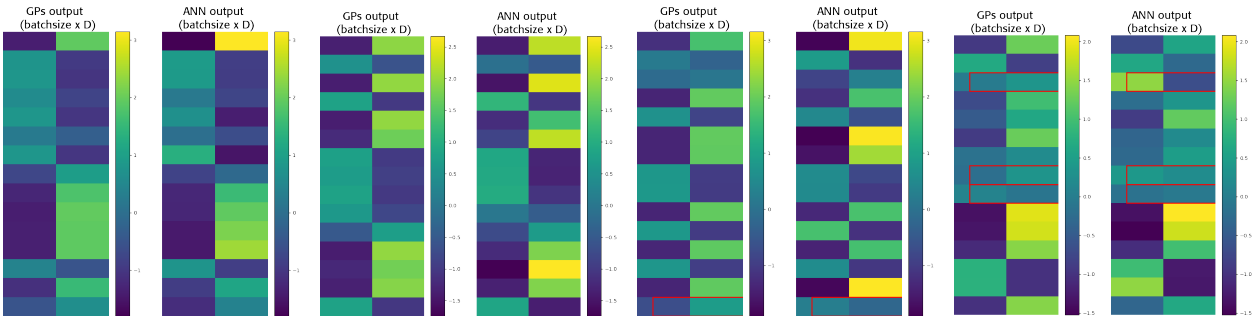
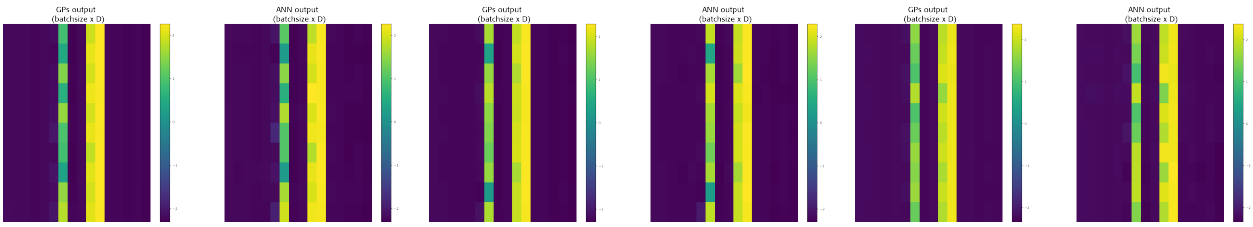
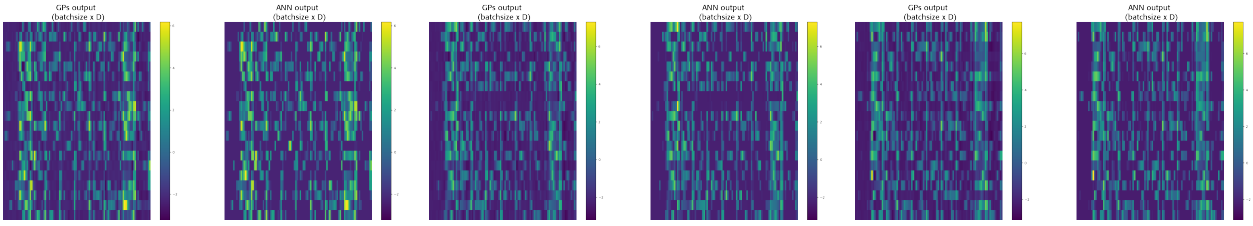


Figure C.7: Comparing GP and ANN outputs for four batches of DogsWolves [131] dataset. The red rectangles highlight the instnaces for which the predictions of GP and ANN (i.e. the class with maximum score) are different.



minibatch 1 minibatch 2 minibatch 3

Figure C.8: Comparing GP and ANN (attention submodule) outputs for 3 batches of Cifar10 [83] dataset.



minibatch 1 minibatch 2 minibatch 3

Figure C.9: Comparing GP and ANN (attention submodule) outputs for 3 batches of Kather [77] dataset.