IN3310 / IN4310 – Deep Learning for Image Analysis

Lecture 8

Performance estimation and generalisation beyond the development dataset

9th of March 2023

Andreas Kleppe (modified by Dhananjay Tomar)



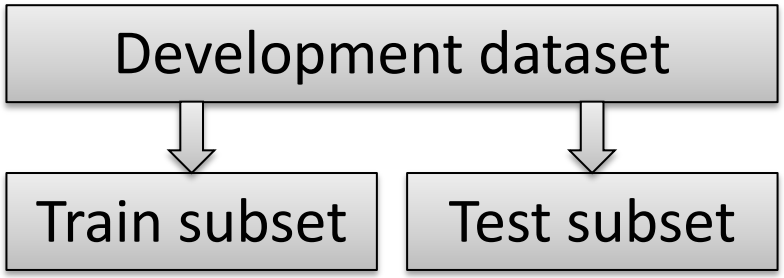
UiO : Department of Informatics
University of Oslo

Progress

- **Internal testing (recap)**
- External testing
- Facilitating generalisation
- Some performance metrics
- Estimator uncertainty

Train and test subsets

- Can randomly partition the development dataset into two disjoint subsets:



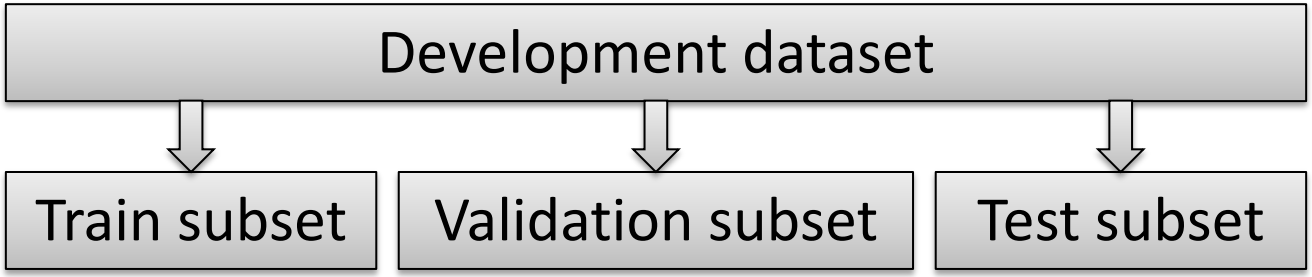
- Use the train subset to train models.
- Use the test subset to evaluate the *final* (ensemble) model.

Train and test subsets

- Unbiased estimation using the development dataset requires:
  - A test subset set aside before starting to experiment.
  - Only a single analysis performed using the test samples.
    - Only a single (ensemble) model.
    - Ideally, also pre-specify what performance metric you would use.
    - It is fine to perform additional analyses but note that these might be biased by the multiple comparisons and learning from previous analyses.
- "if you don't like the results, you have to obtain, and lock away, a completely new test set if you want to go back and find a better hypothesis."
  - Page 709 in Russell, S. & Norvig, P. *Artificial Intelligence: A Modern Approach* 3rd edn (Prentice Hall, 2010).
- Sadly, this is too often ignored.

Validation subset

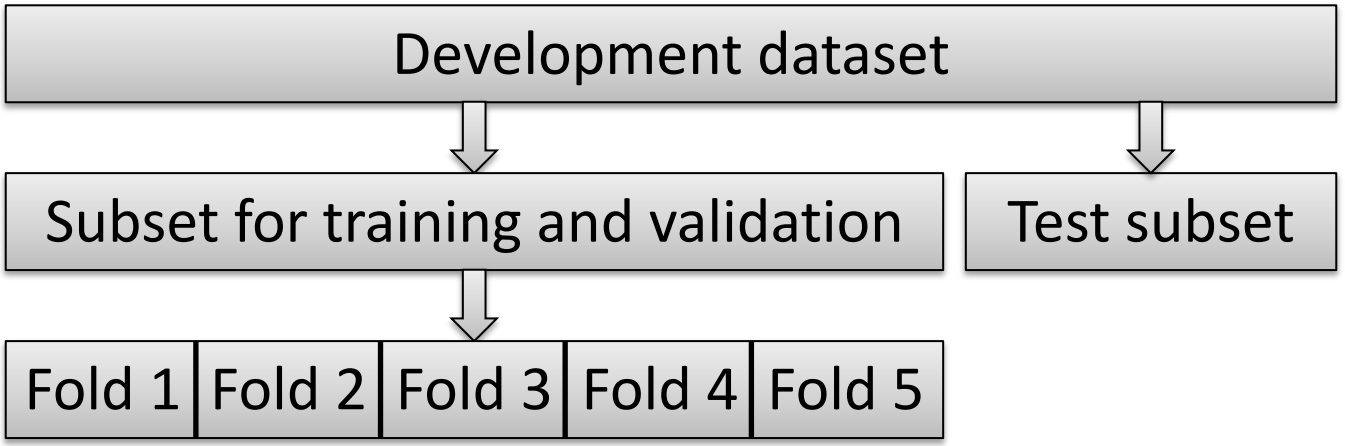
- A third disjoint subset of the development dataset can be used to evaluate different modelling choices.
  - "Modelling choices" means all hyperparameters, including choices of methods such as type of neural network and optimisation method.



- Nomenclature note: Some use "validation" for testing and "tune subset" as the subset for evaluating modelling choices.
  - Should not be confused with fine-tuning, i.e. training from an initialisation that applies weights from a pre-trained model.

Validation by resampling

- Instead of a fixed validation subset, we can make multiple validation subsets and use the average performance to evaluate modelling choices.
- *n*-fold cross-validation:
  1. Randomly partition the non-test subset into *n* folds of equal size.
  2. Repeat *n* time:
     Use the n^{th} fold for validation, and the other *n*-1 folds for training.
  3. Use the average of the performance estimates obtain for the validation folds.
     - Could also calculate the standard deviation or other measures of uncertainty.



Validation by resampling

Pros:

- Estimates are less influenced by one particular training.
- Estimates are less influenced by one particular validation subset.
- Ends up with *n* models which can be used in an ensemble model.

Con:

- Training time (and validation time) multiplied by *n*
  - Could be acceptable for reasonably small *n*, e.g., 5.

Validation by resampling

- *n*-fold cross-validation applies a fixed partitioning into folds.
- We can instead repeatedly partition the non-test subset into a train subset and a validation subset. For e.g., using bootstrapping.
- Bootstrapping example: repeat 10 times:
  1. Sample with replacement from the non-test data to obtain a train subset with the same size as the non-test subset.
     - Each train subset will on average contain 63.2% of the non-test data but will be of the same size as the non-test subset due to duplicates.
  2. Train using the train subset (with duplicates), and use the other non-test data for validation.

Validation

- Using resampling techniques for validation will use the data more effectively than applying a fixed validation subset.
  - Because all non-test data will be used for both training and validation.
- Evaluating modelling choices multiple times will make the best performance estimates overly optimistic.
  - Due to multiple comparisons and learning from previous analyses.
  - With a fixed validation subset, the choice that appears to provide best performance might do so only for the particular validation subset.
  - When using a resampling technique, the choice that appears to provide best performance might do so only for the non-test data.
- If data is limited, applying a resampling technique is usually preferable if the added computational expense is acceptable.

Validation

- Example from classical machine learning and medicine:
- ≈20 years ago, microarray gene-expression profiling was the new hot.
- Michiels et al. used a resampling technique to re-evaluate publicly available data from seven studies that had applied specific validation subsets to evaluate the performance of trained molecular signatures.
  - The studies were published in Nature, Nature Medicine, Nature Genetics, Lancet, NEJM, PNAS, and Cancer Cell.
- They found that "Five of the seven studies did not classify patients better than chance."

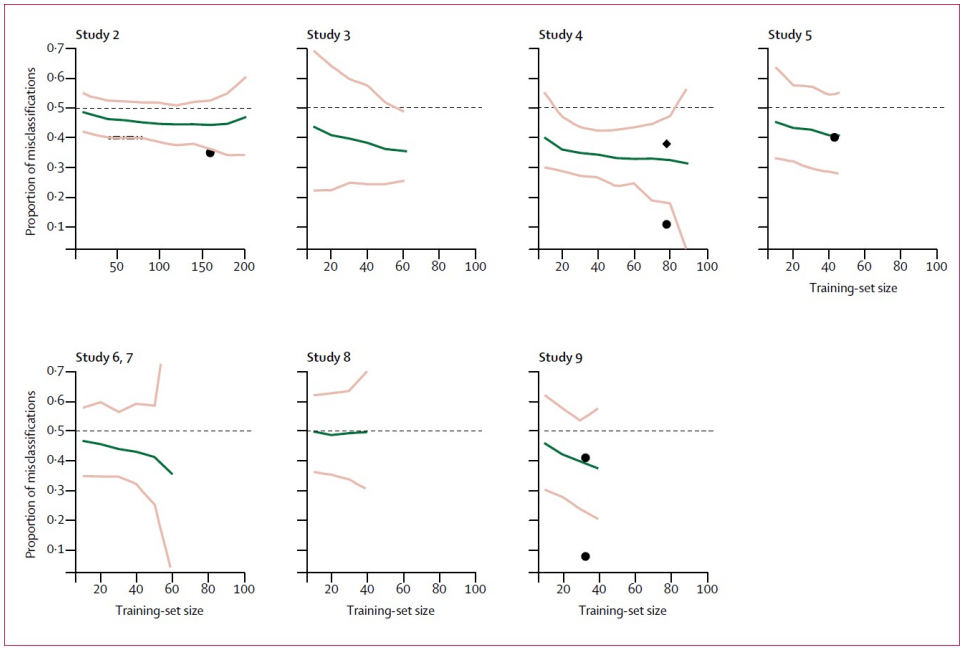


Figure 2: **Proportion of misclassifications in validation sets as a function of corresponding training-set sizes in the seven studies**[2–9]
Green lines=mean proportion of misclassifications obtained from 500 random training-validation sets as a function of the training-set size. Pale red lines=95% CIs. Dots=misclassification rates in original publications. Iizuka and colleagues[9] published two misclassification rates by two different methods on the same validation set. Diamond=second misclassification rate on a larger independent validation set[16] from the van 't Veer study.[4]

Michiels, S., Koscielny, S. & Hill, C. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *Lancet* **365**, 488–492 (2005).

Subset sizes

- Common to randomly partition the development data using some percentages, e.g., 70% for training, 15% for validation, and 15% for testing.
- Resampling techniques might allow use of less validation data.
- Size of test subset should be considered in relation to the uncertainty of the resulting estimates.
  - If performance differences of <1 percentage point should be considered, then thousands of test samples are usually needed.

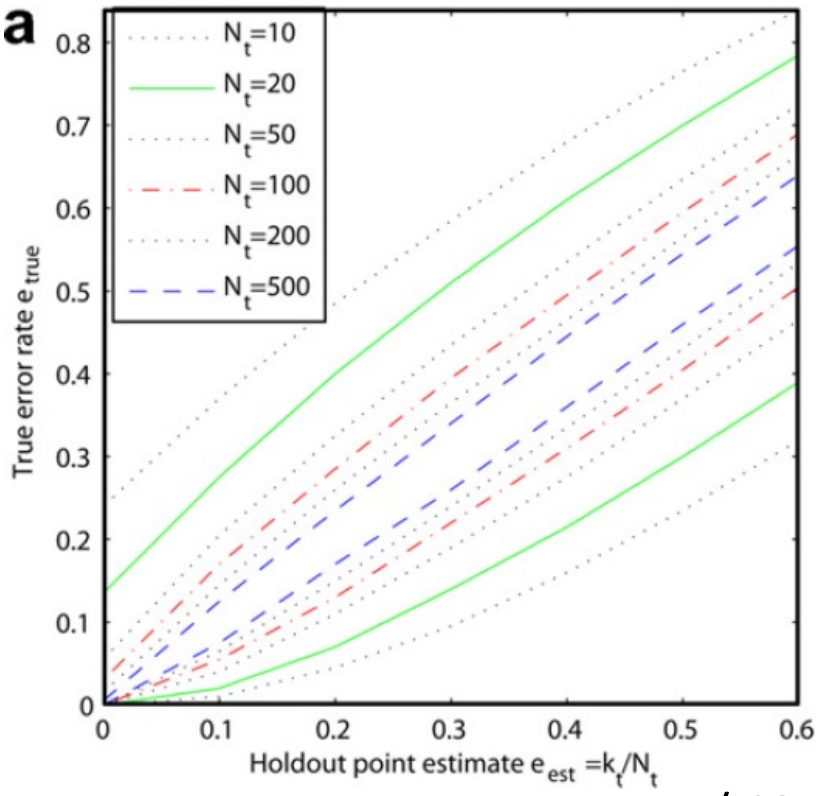


Fig: The lines depict the range of 95% Bayesian confidence intervals.

Figure 4a by Isaksson, A., Wallman, M., Göransson, H. & Gustafsson, M. G. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognit. Lett.* **29**, 1960–1965 (2008).

Subset sizes

- The relation between performance and size of training data appears to be linear on log-log scale.
  - Thus, the performance gained from increasing the size of the train subset is initially large and then quickly becomes smaller.
  - With too small train subset, the model struggles to learn.
  - With an excessive train subset, adding more training data might not improve the performance.

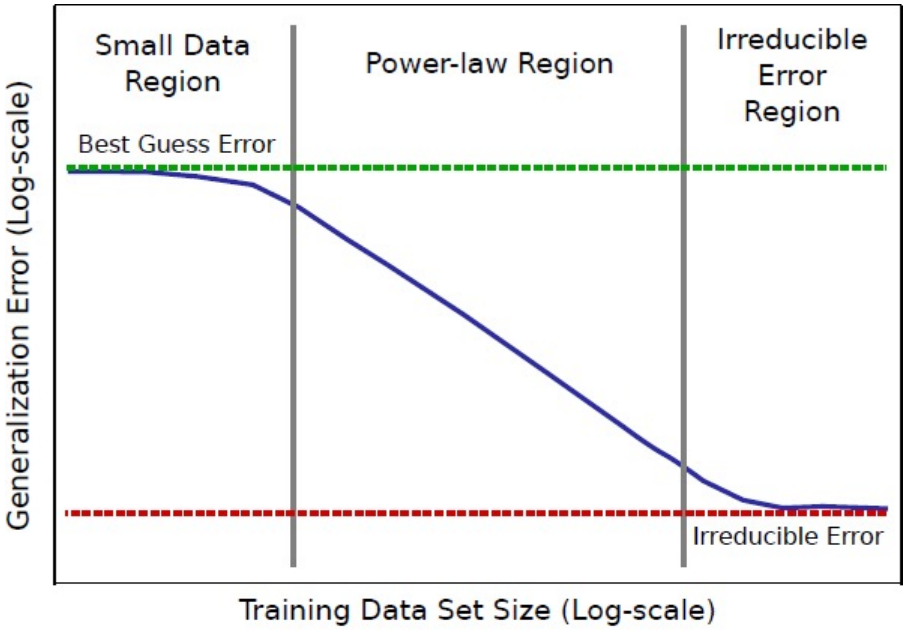


Figure 6: Sketch of power-law learning curves

Hestness, J. et al. Deep learning scaling is predictable, empirically. https://arxiv.org/pdf/1712.00409.pdf (2017).

Progress

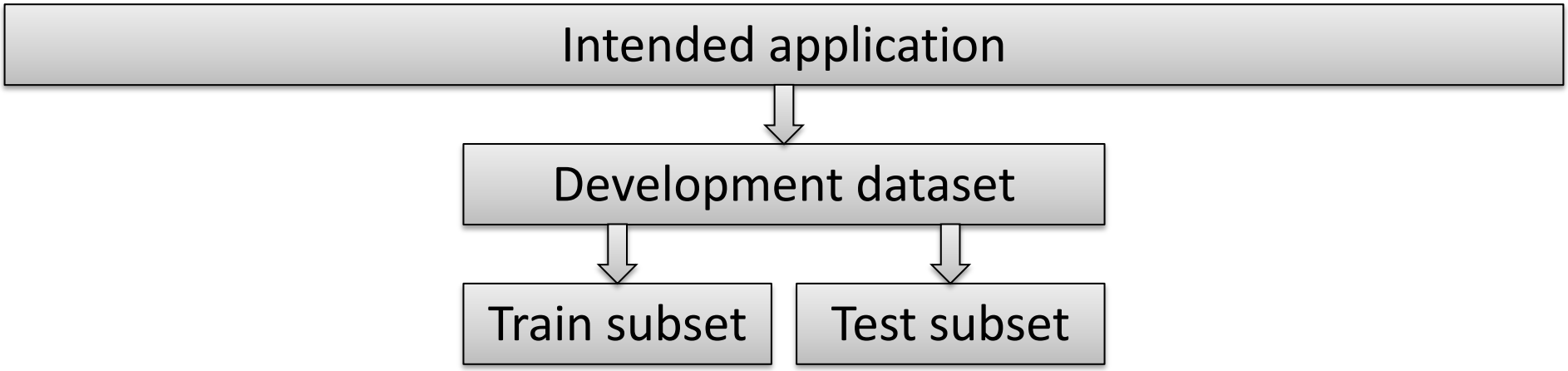
- Internal testing (recap)
- **External testing**
- Facilitating generalisation
- Some performance metrics
- Estimator uncertainty

Representability

- Recap: If a random test subset is used for only a single analysis, then the resulting estimate is expected to be unbiased for the development data.
- Does that imply unbiased estimation for the intended application?
- No, not if the development dataset is not representative of the intended application.
- Example: Detecting values of coins
  - Intended application: all coins (with and without scratches).
  - Development dataset: coins without scratches.
  - Consequence: performance on test subset might be good, but performance in the intended application might be poor.

Representability

- We often assume that the development dataset is a set of realisations of independent and identically distributed (iid) random variables representative of the intended application.
- Unfortunately, this is rarely true.
- Instead, the development dataset is often fairly representative of one or a few historical settings where we would like our model to work.



Domain Shift

Training Data





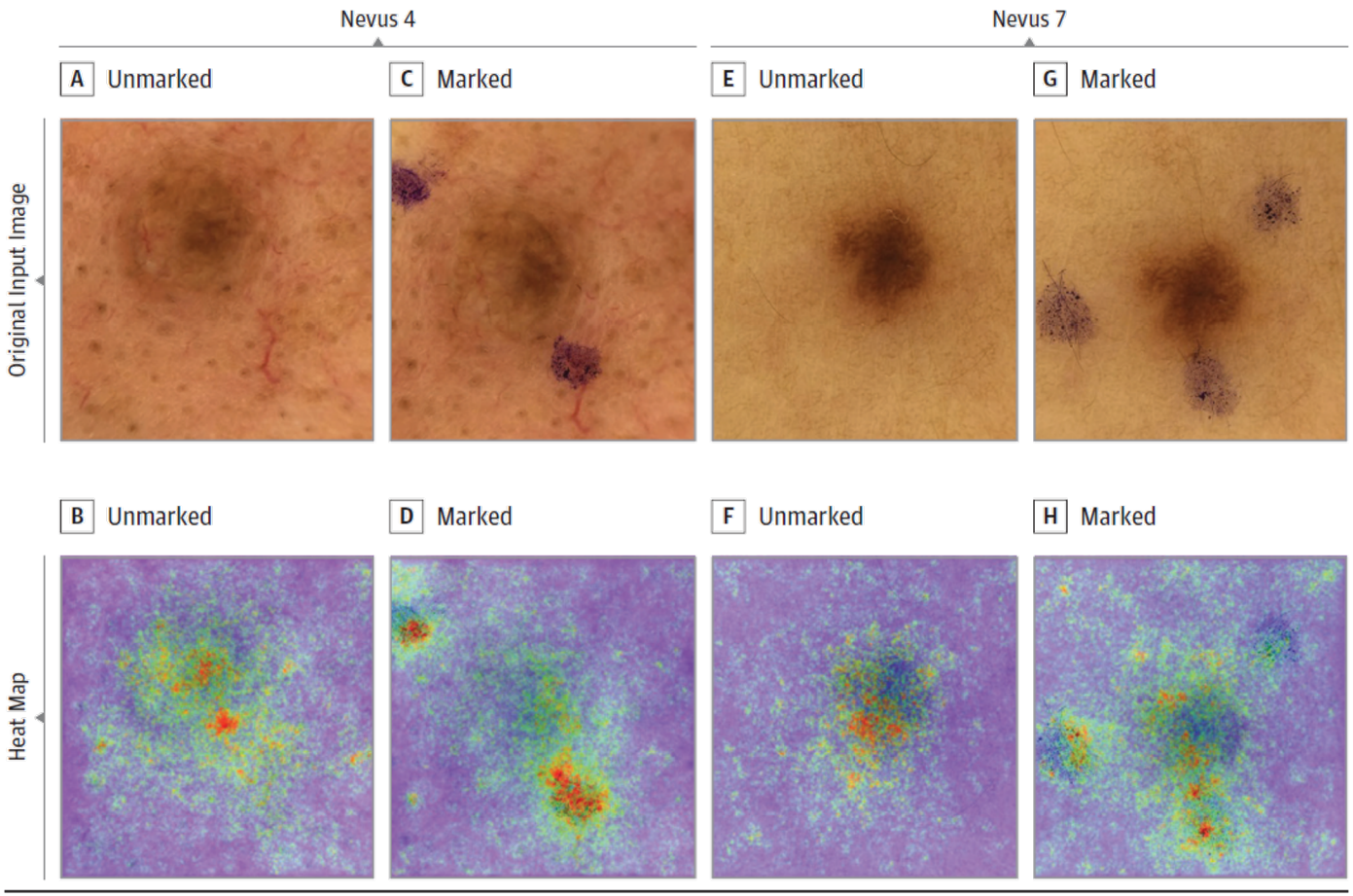
Real world





Representability

Figure 4. Heat Maps of 2 Benign Nevi With Major Increase in Melanoma Probability Scores After Addition of In Vivo Skin Markings



The heat maps were created by vanilla (meaning basic) gradient descent backpropagation. A and E, Unmarked input images. B and F, Heat maps reveal relevant pixels for the convolutional neural network's (CNN's) prediction of benign nevi. C and G, Marked input images. D and H, Heat maps reveal that skin markings are of high relevance for CNN's changed prediction of malignant melanomas, while the nevus itself is mostly ignored.

Winkler, J. K. et al. Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition. *JAMA Dermatol.* **155**, 1135–1141 (2019).

Representability

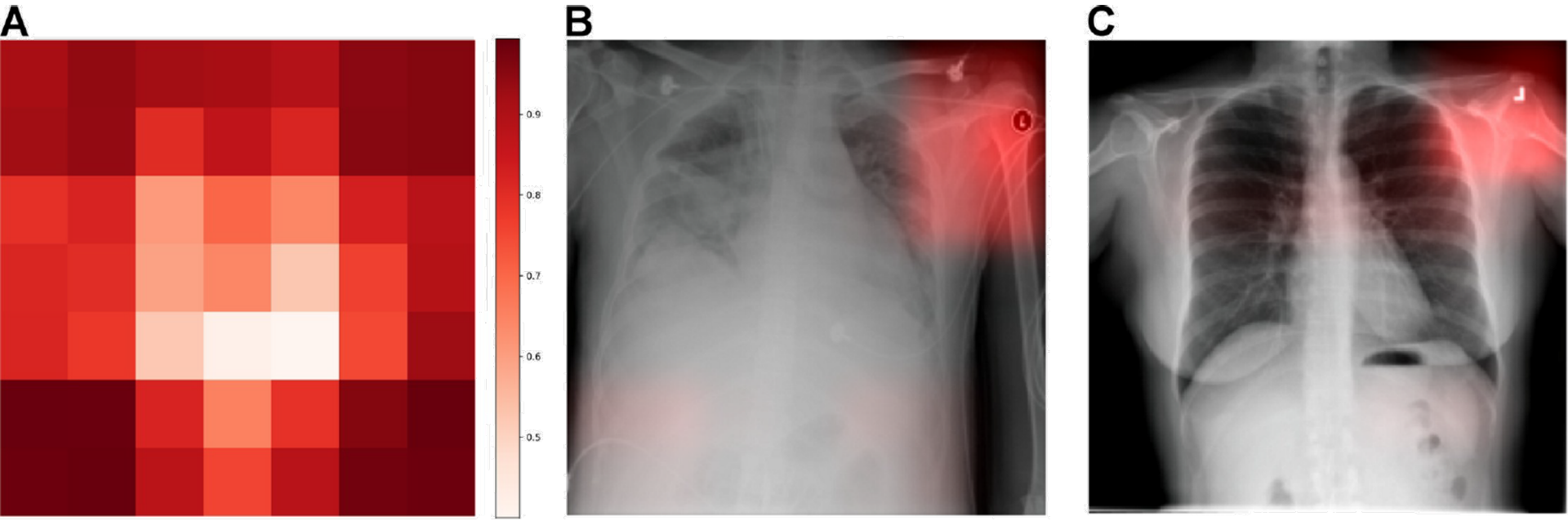
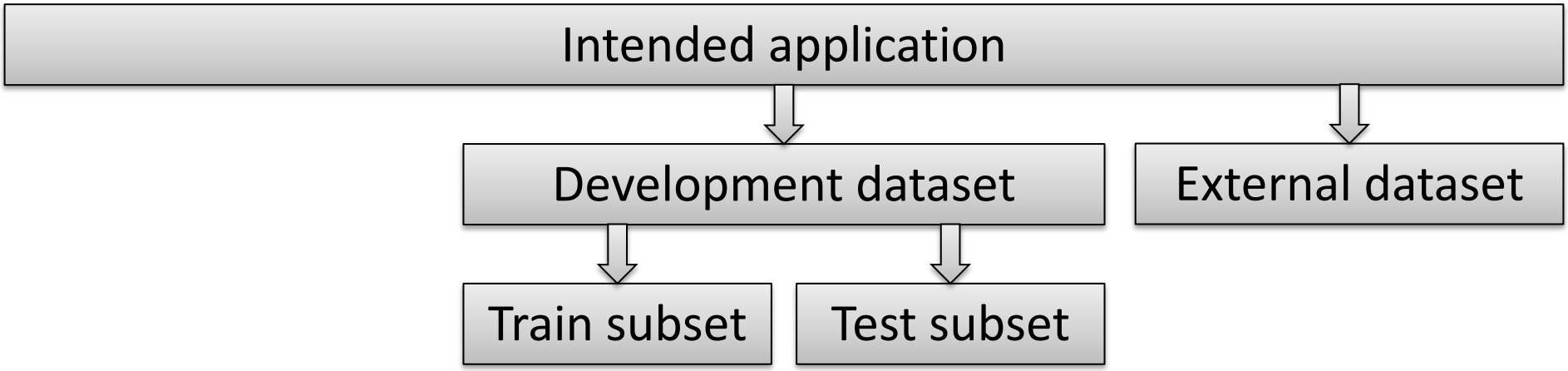


Fig 2. CNN to predict hospital system detects both general and specific image features. (A) We obtained activation heatmaps from our trained model and averaged over a sample of images to reveal which subregions tended to contribute to a hospital system classification decision. Many different subregions strongly predicted the correct hospital system, with especially strong contributions from image corners. (B-C) On individual images, which have been normalized to highlight only the most influential regions and not all those that contributed to a positive classification, we note that the CNN has learned to detect a metal token that radiology technicians place on the patient in the corner of the image field of view at the time they capture the image. When these strong features are correlated with disease prevalence, models can leverage them to indirectly predict disease. CNN, convolutional neural network.

Zech, J. R. et al. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross- sectional study. *PLoS Med.* **15**, e1002683 (2018).

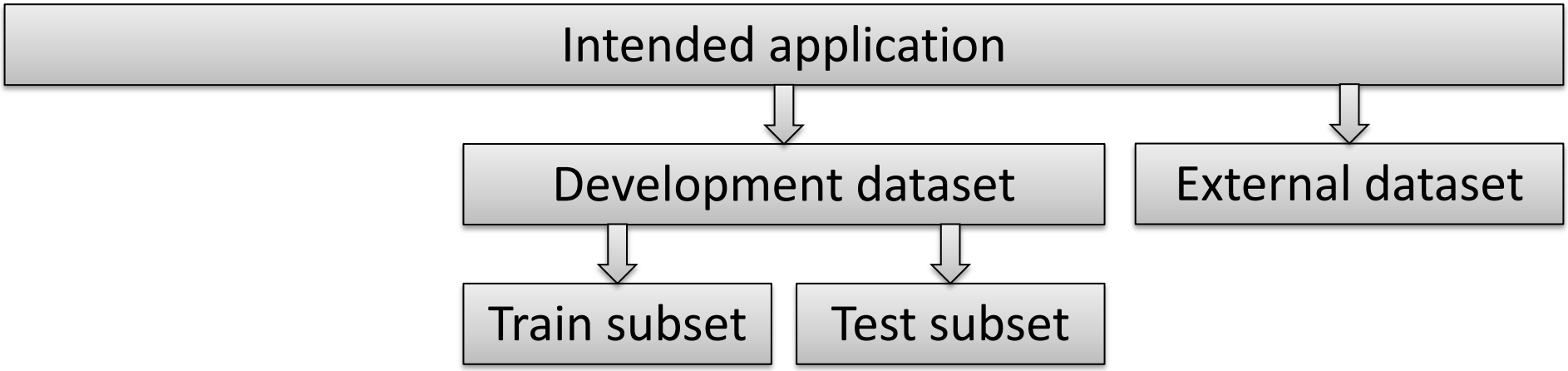
Representability

- Data external to the development dataset are needed to obtain more realistic performance estimates.
  - What is considered “external” varies a bit, for e.g., being from different geographical locations than the development dataset.
  - A dataset temporally different (e.g. from a later time period) is sometimes considered external.
  - Note: External datasets are a luxury that is too seldom available.



Representability

- Also, an external dataset is usually not fully representative of the intended application.
- Good performance on an external dataset:
  - Indicates good generalisation to one of multiple intended settings.
    - Assuming the external dataset is representative of an intended setting.
  - Suggests that the model would also generalise to other settings.



Use of external datasets

- Performance estimated using an external dataset becomes overoptimistic if multiple models are evaluated and the best one selected.
  - Analogous to multiple testing on a test subset.
- The external dataset should therefore be used only once to evaluate the final model.
  - Details related to the evaluation should ideally also be pre-specified, e.g., precisely which samples and performance metric.
  - Additional analyses could be performed (using e.g., other models, subsets of samples, or other metrics), but those estimations should not be considered unbiased.
    - Due to multiple comparisons and learning from previous analyses.

Use of external datasets

- External datasets are becoming more common in applied deep learning studies.
- Example from cancer diagnostics:

Statistics for studies with ≥20 citations per year or published in journals with impact factor ≥10.

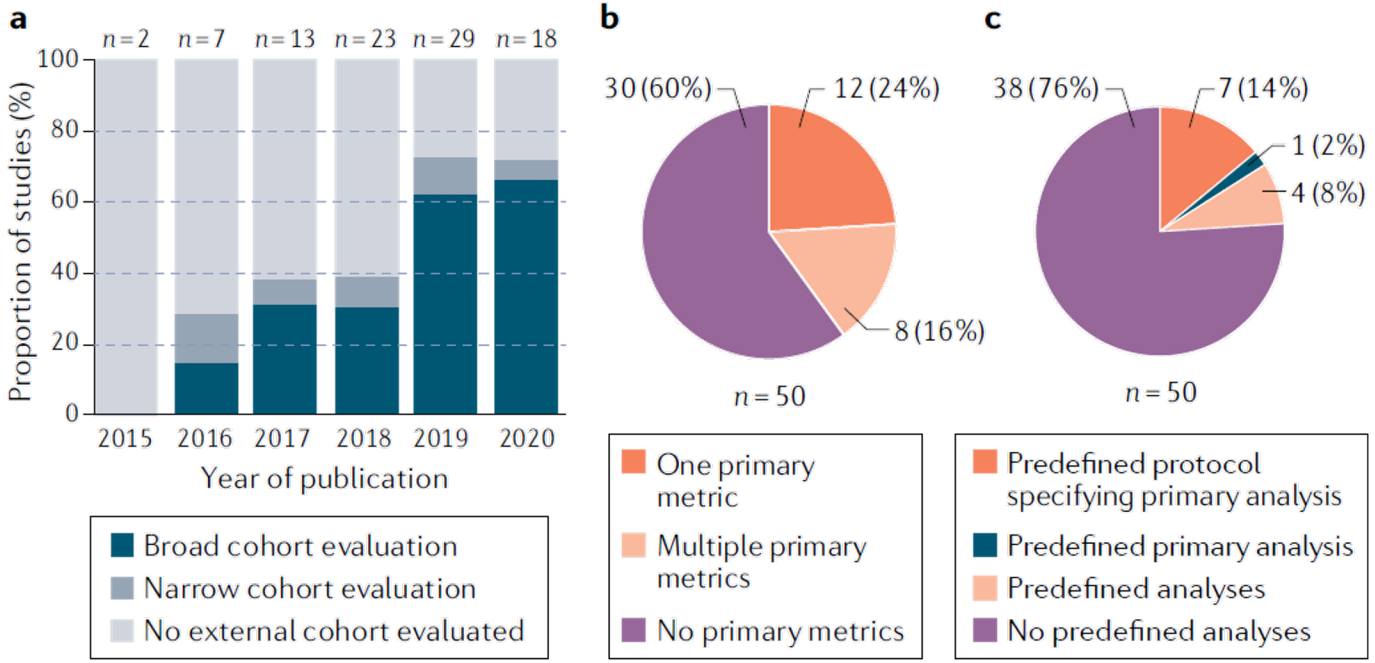


Figure 1 in Kleppe, A. et al. Designing deep learning studies in cancer diagnostics. *Nat. Rev. Cancer* **21**, 199–211 (2021).

- Pre-specification of the external test is rarely reported.
  - Except in clinical trial studies, where pre-specification is usually required and publicised in advance of patient recruitment.

Progress

- Internal testing (recap)
- External testing
- **Facilitating generalisation**
- Some performance metrics
- Estimator uncertainty

Facilitating generalisation

Control the network's capacity.

- Reduce the number of trainable parameters.
  - Stacking convolutional layers with small kernel sizes.
  - Depthwise separable convolutions.
  - Reduce width (e.g., number of channels in convolutional layers) and depth (number of layers).
    - Should be scaled appropriately with input size (e.g., image resolution), e.g., as in EfficientNet.
- Dropout.
- Weight decay.

Facilitating generalisation

- Control the network's capacity.
- Facilitate learning.
  - Residual connections.
  - Batch normalisation.
  - Transfer learning.
  - Learning rate schedule.
  - Optimisation method.

Facilitating generalisation

- Control the network's capacity.
- Facilitate learning.
- Image-specific normalisation.
  - Aims at making different images more similar.
  - Differs from dataset-based input normalisation such as subtraction of channel-wise mean and division by the channel-wise standard deviation with mean & std estimated for the training data.
- Data augmentation.
  - E.g., randomly distort brightness, colour, contrast, rotation, sharpness, or size of each input image during training.
  - See e.g., Shorten, C. & Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *J. Big Data* **6**, 60 (2019).
- More diverse training data.

Facilitating generalisation

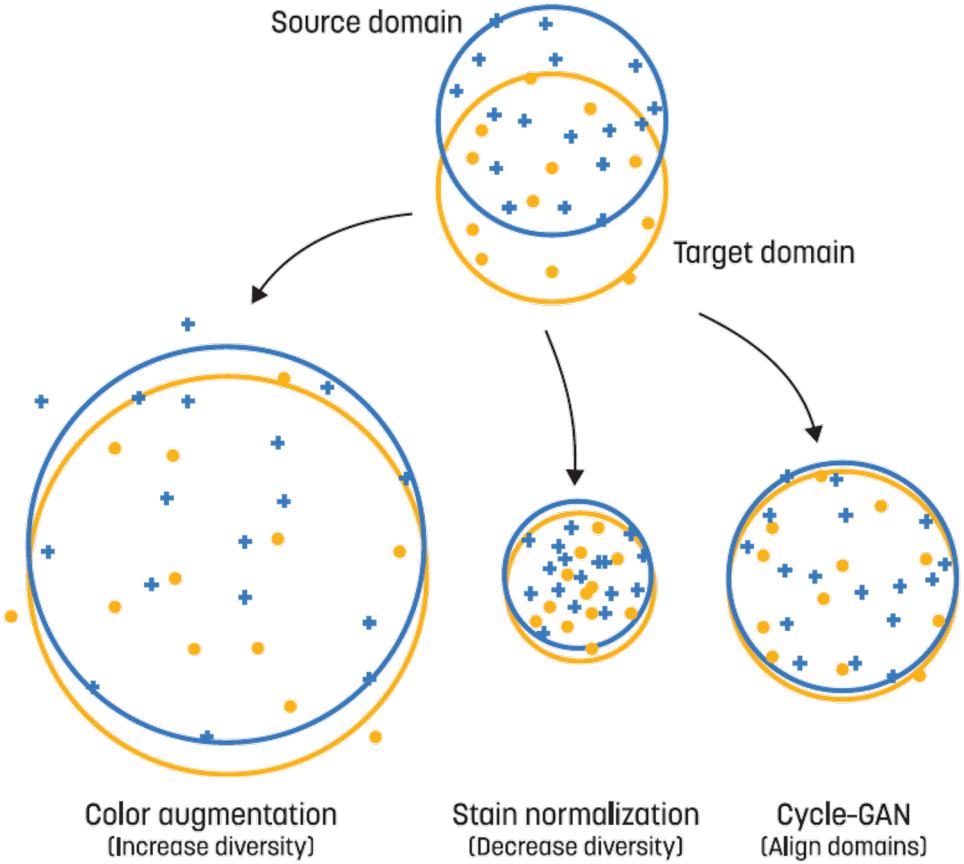


Fig. 6. Data preparation aim at increasing the intersection between domains. Different data transformation techniques do this with different strategies.

Stacke, K., Eilertsen, G., Unger, J. & Lundström, C. Measuring domain shift for deep learning in histopathology. *IEEE J. Biomed. Health Inform.* **25**, 325–336 (2021).

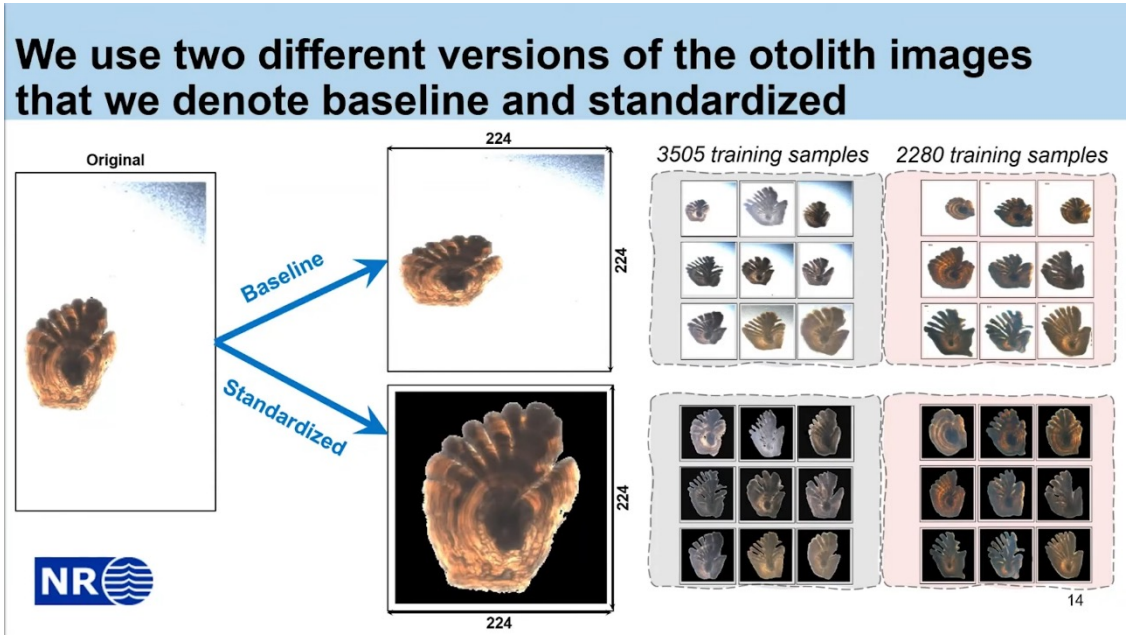
We are not going to focus on domain adaptation approaches (e.g. using Cycle-GAN) because it generally requires learning adaptation using multiple samples from the target domain, which might severely limit the practical applicability of the prediction system.

Normalisation

- Dataset-based input normalisation can facilitate faster training convergence, possibly leading to a better model.
  - All images are transformed using the same function and parameters.
- Image-specific normalisation can make images more similar.
  - Each image is usually processed by a fixed algorithm, but this algorithm can apply parameters extracted from that particular image.
  - Simple example: tf.image.per_image_standardization
    1. Compute mean and standard deviation of all values of a particular image.
    2. Subtract mean from each value of the particular image and divide by the standard deviation (capped away from 0).
       - Sounds very similar to dataset-based input normalisation, but note that the mean and standard deviation is computed **for the particular image and not across all training images**.
       - Similar image-specific normalisation was previously common to apply before extracting textural features.

Normalisation

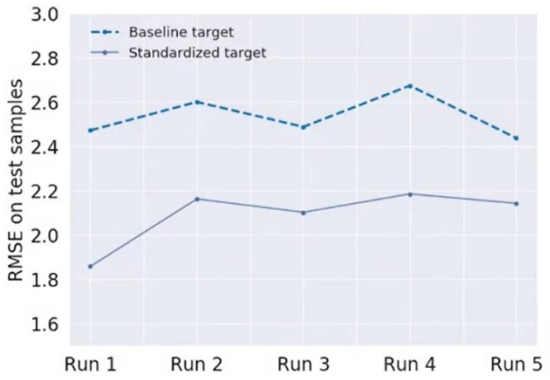
- Another example of image-specific normalisation is cropping and centre alignment of region of interest.



153 test samples

Target domain	Lower Bound	CoGAN (Average over 5 runs)	Upper Bound
Baseline	5.0319	2.5332 (74.5%)	1.6764
Standardized	5.8589	2.0889 (93.6%)	1.8311

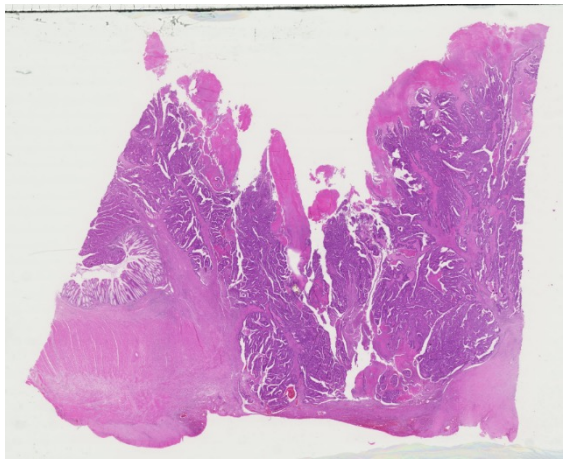
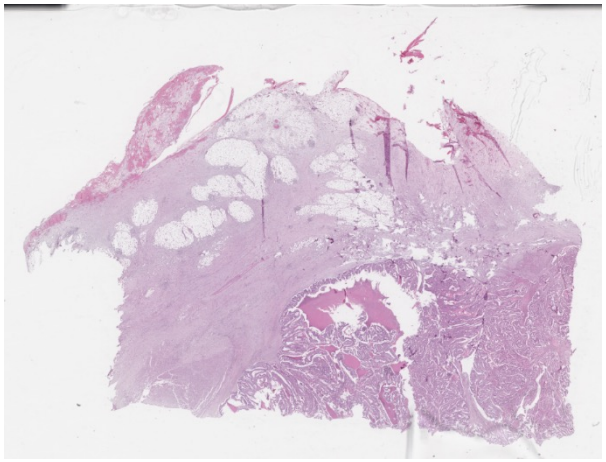
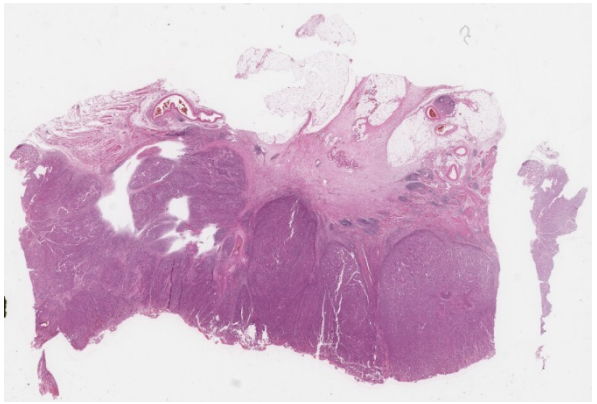
RMSE (year)



Ordoñez, A. Deep Domain Adaptation Applied to Fish Age Prediction. *Visual Intelligence Seminar* (2021).
https://www.youtube.com/watch?v=5NwGGsHyqbA

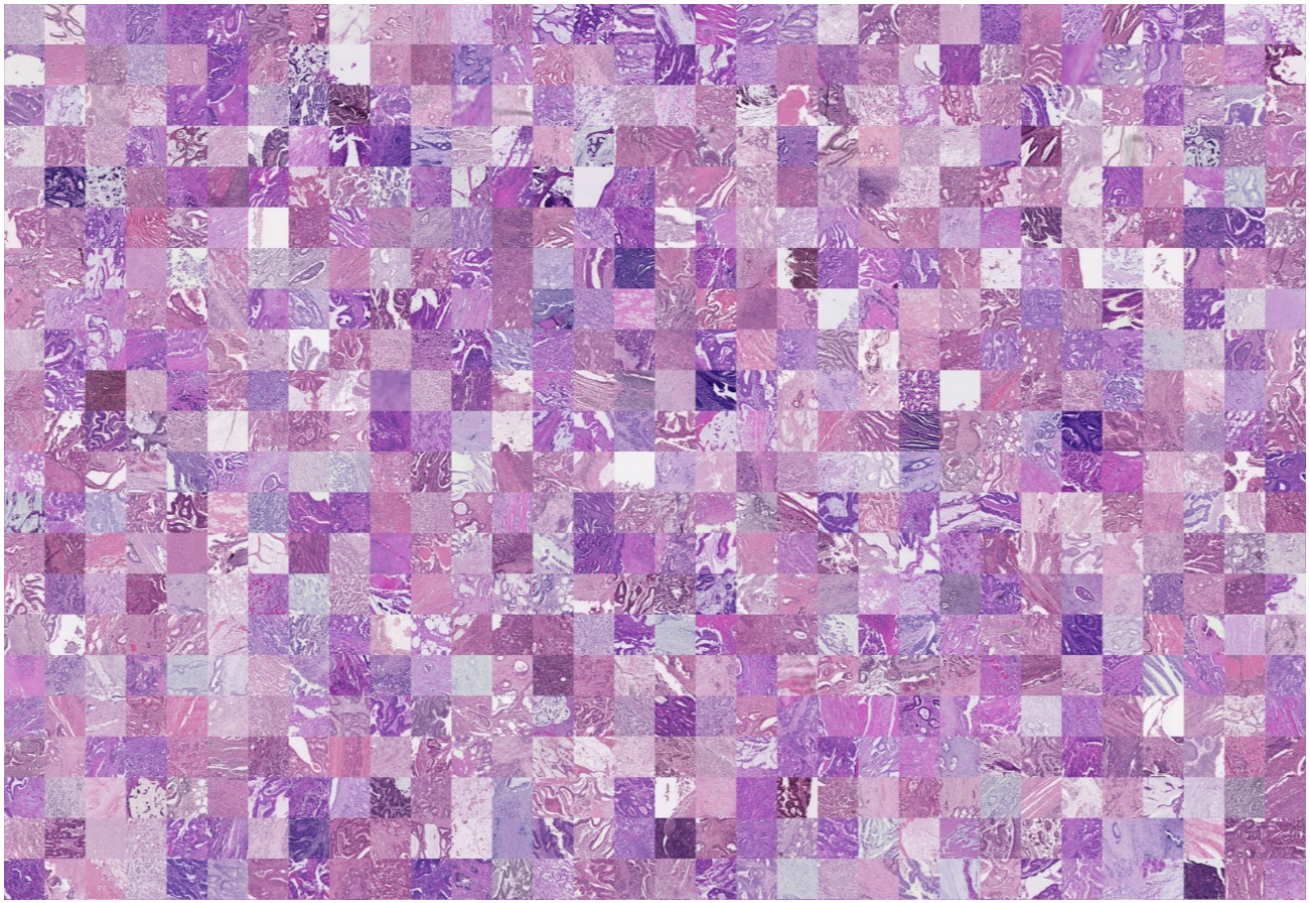
Normalisation

- There also exists more sophisticated image-specific normalisation methods.
- Consider the example of images of tissue sections stained with the conventional histopathological staining (called H&E).
- Differences in fixation of tissue, sectioning, staining, and scanner devices can all result in substantial image variation:

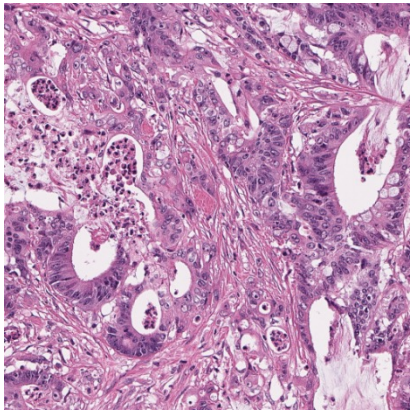
Normalisation

- The large variation is perhaps even more apparent when comparing different image patches extracted from the scans:

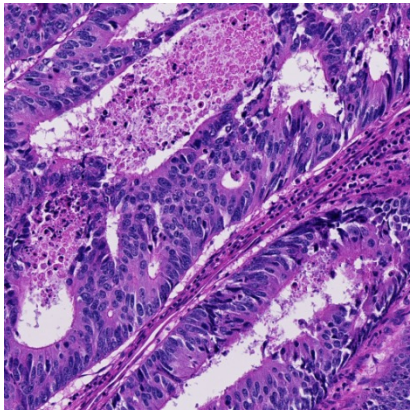


Normalisation

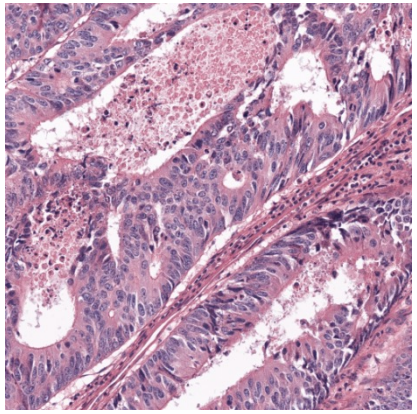
- Macenko normalisation is an image-specific normalisation method for such images.
- Main principle: Only two stains are applied to the tissue (Haematoxylin and Eosin), so it should be possible to detect basis vectors representing these colours and swap those for a source image with those for a target image.
- Also the relative concentration of each stain in the source image should be normalised to match the relative concentration of each stain in the target image.



Target



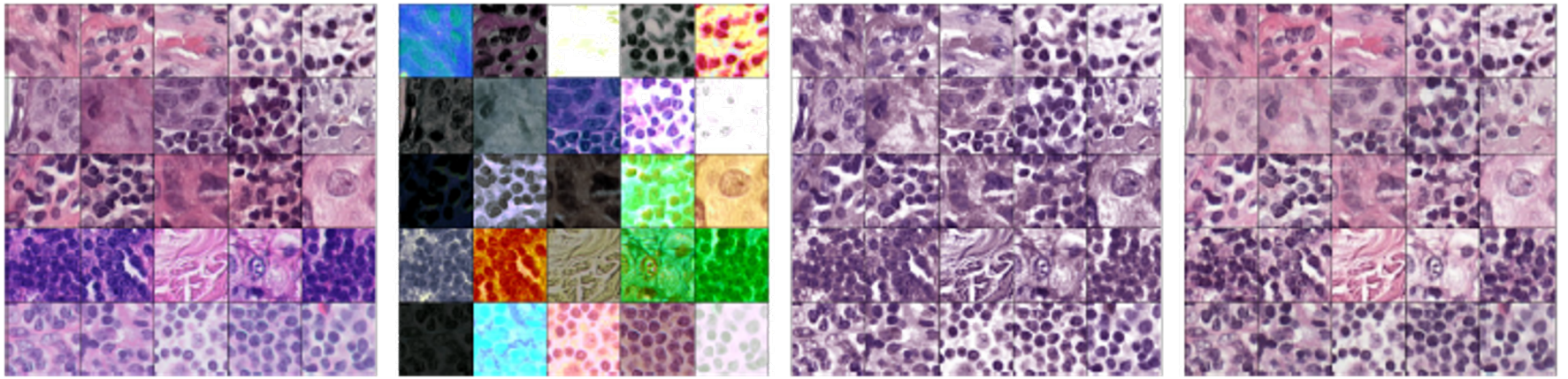
Source



Source normalised



Example for tumour detection



(a) Original (b) Color augmentation (c) Stain normalization (d) Cycle-GAN

Fig. 5. Five example images per center from the CAMELYON17 dataset (total 5x5 images), shown with different transformations. The three top rows per image are scanned with Scanner 1, the forth row with Scanner 2 and the last row with Scanner 3.

Stacke, K., Eilertsen, G., Unger, J. & Lundström, C. Measuring domain shift for deep learning in histopathology. *IEEE J. Biomed. Health Inform.* **25**, 325–336 (2021).

Example for tumour detection

TABLE I

CLASS BALANCED MEAN ACCURACIES (STANDARD DEVIATION) (%), TRAINING A TUMOR CLASSIFIER ON SCANNER 1 FOR CAMELYON17, AND CENTER 1 FOR AIDA-LNCO, TESTING IT ON SAME-SCANNER/CENTER DATA AND OTHER DATA, USING TWO DIFFERENT MODEL ARCHITECTURES, EVALUATED ACROSS 5 INDEPENDENT TRAINING RUNS

CAMELYON17

	Simple CNN				*Mini-GoogLeNet*			
Train \ Test	Scanner 1	Scanner 2	Scanner 3	Mean drop (p.p.)	Scanner 1	Scanner 2	Scanner 3	Mean drop (p.p.)
Orig. data	90.0 (5.0)	48.9 (1.2)	72.3 (12.4)	29.5 (9.7)	95.4 (2.5)	49.7 (3.4)	73.8 (12.8)	33.6 (7.4)
Color aug.	94.5 (1.3)	91.4 (0.8)	89.3 (3.9)	4.1 (1.7)	96.4 (0.7)	90.3 (0.2)	93.5 (1.1)	4.5 (0.5)
Stain norm.	94.1 (2.1)	93.5 (1.3)	92.7 (2.0)	0.9 (1.5)	97.8 (1.0)	93.5 (2.5)	96.5 (0.7)	2.8 (1.3)
Cycle-GAN	95.0 (1.3)	91.0 (1.7)	91.8 (0.6)	3.6 (0.8)	97.0 (0.7)	90.1 (2.5)	94.4 (0.4)	4.7 (1.6)

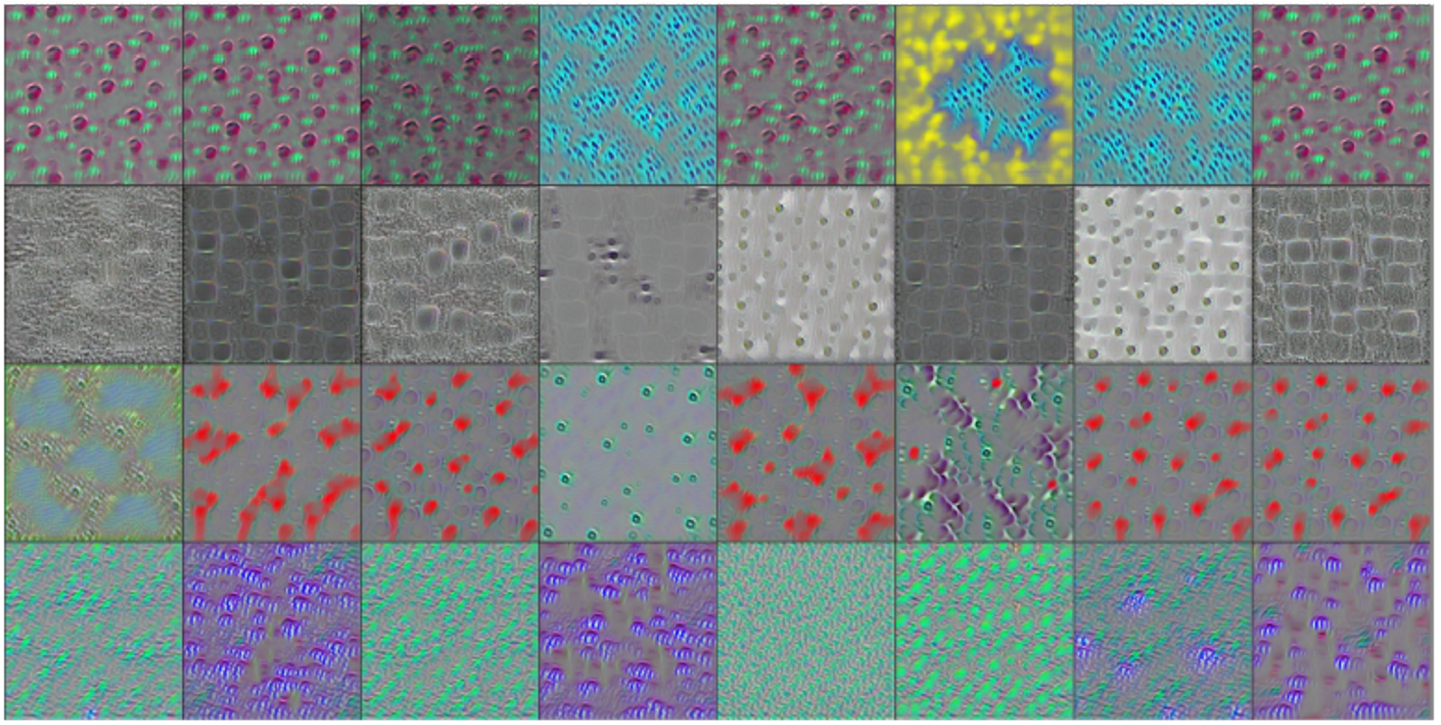
AIDA-LNCO

	Simple CNN			*Mini-GoogLeNet*		
Train \ Test	Center 1	Center 2	Mean drop (p.p.)	Center 1	Center 2	Mean drop (p.p.)
Orig. data	98.3 (0.8)	73.3 (6.6)	25.0 (5.8)	97.3 (1.3)	72.8 (8.1)	24.5 (7.5)
Color aug.	96.0 (3.3)	95.8 (1.0)	0.3 (2.5)	95.3 (4.6)	95.5 (2.0)	−0.2 (2.7)
Stain norm.	96.3 (4.0)	95.5 (3.1)	0.8 (0.9)	95.4 (3.7)	95.7 (2.6)	−0.3 (2.0)
Cycle-GAN	98.3 (1.2)	77.7 (3.4)	20.6 (3.6)	97.2 (1.3)	81.7 (10.7)	15.5 (10.5)

Stacke, K., Eilertsen, G., Unger, J. & Lundström, C. Measuring domain shift for deep learning in histopathology. *IEEE J. Biomed. Health Inform.* **25**, 325–336 (2021).

- For generalisation to external data, it seems that colour augmentation helps primarily because it increases the variability of the input, not because it artificially increases the amount of training data.

Example for tumour detection



(b) Mini-GoogLeNet, the convolutional layer in the first auxiliary branch

Fig. 2. Each row shows example images that maximally activates different filters in models Simple CNN (a) and Mini-GoogLeNet (b), trained with (from top to bottom): original data, color augmentation, stain normalization, and Cycle-GAN. Best viewed digitally.

Stacke, K., Eilertsen, G., Unger, J. & Lundström, C. Measuring domain shift for deep learning in histopathology. *IEEE J. Biomed. Health Inform.* **25**, 325–336 (2021).

Augmentation + normalisation?

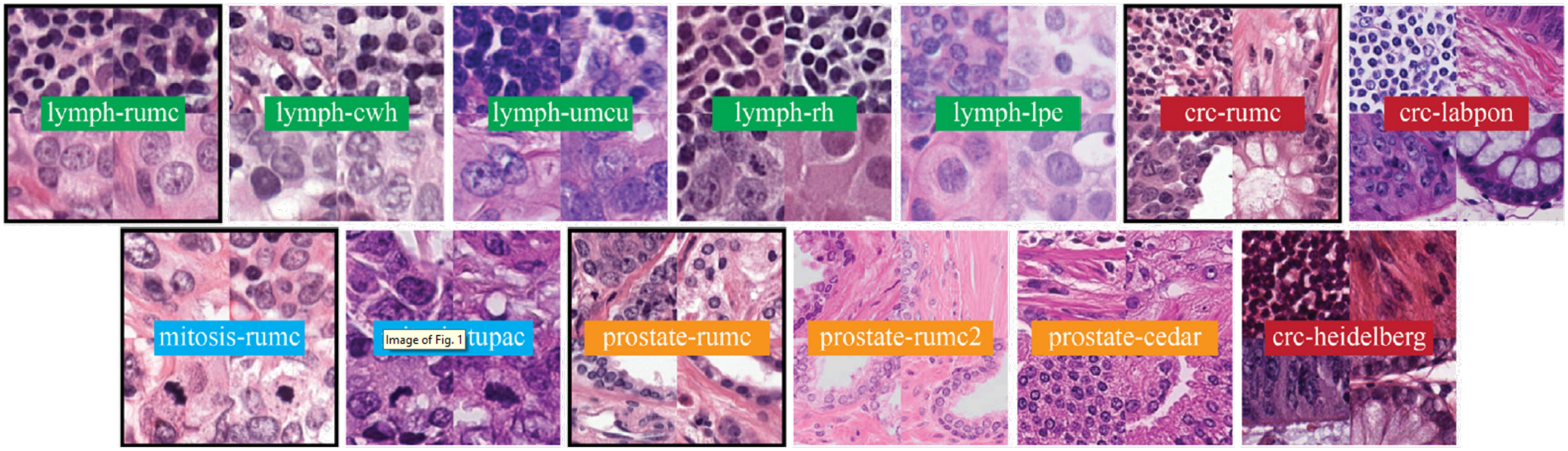


Fig. 1. Example images from training and test datasets. Applications are indicated by colors and keywords: tumor detection in lymph nodes (*lymph*), colorectal cancer tissue classification (*crc*), mitosis detection (*mitosis*) and prostate epithelium detection (*prostate*). Training set images are indicated by the keyword *rumc* and black outline. The rest belong to test sets from other centers. Stain variation can be observed between training and test images.

Tellez, D. et al. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Med. Image Anal.* **58**, 101544 (2019).

Augmentation + normalisation?

Table 1
Experimental results ranking *stain color augmentation* and *stain color normalization* methods. Values correspond to AUC scores, except for the last column, averaged across 5 repetitions with standard deviation shown between parenthesis. Each column represents a different external test dataset, with the last column *Ranking* indicating the position of each method within the global benchmark, computed as described in Section 4.1. Normalization methods: *Network* is our proposal; *Style* is from Bug et al. (2017); *LUT* is from Bejnordi et al. (2016); and *Deconvolution* is from Macenko et al. (2009).

Normalization	Augmentation	lymph-cwh	lymph-lpe	lymph-rh	lymph-umcu	mitosis-tupac	prostate-rumc2	prostate-cedar	crc-labpon	crc-heidelberg	Ranking
Identity	HED-light	0.952(0.004)	0.976(0.001)	0.946(0.009)	0.968(0.004)	0.996(0.001)	0.957(0.001)	0.879(0.011)	0.973(0.002)	0.895(0.002)	1.2(0.4)
Style	HED-light	0.961(0.002)	0.953(0.004)	0.952(0.001)	0.972(0.004)	0.991(0.003)	0.925(0.003)	0.879(0.006)	0.975(0.001)	0.917(0.003)	2.8(0.7)
Network	HSV-light	0.946(0.006)	0.962(0.001)	0.941(0.002)	0.965(0.004)	0.992(0.001)	0.957(0.000)	0.872(0.013)	0.980(0.001)	0.900(0.003)	3.9(1.9)
Network	HED-light	0.949(0.005)	0.968(0.001)	0.942(0.002)	0.963(0.004)	0.989(0.003)	0.958(0.001)	0.862(0.011)	0.980(0.001)	0.906(0.003)	4.1(1.6)
Identity	HSV-strong	0.955(0.003)	0.965(0.004)	0.929(0.002)	0.973(0.003)	0.988(0.003)	0.945(0.009)	0.886(0.005)	0.977(0.001)	0.902(0.003)	4.7(1.7)
Network	HSV-strong	0.953(0.002)	0.964(0.003)	0.946(0.002)	0.964(0.005)	0.991(0.003)	0.951(0.002)	0.852(0.006)	0.975(0.002)	0.894(0.005)	6.6(0.9)
Network	HED-strong	0.956(0.003)	0.959(0.002)	0.940(0.003)	0.965(0.004)	0.985(0.005)	0.943(0.003)	0.861(0.009)	0.974(0.002)	0.916(0.003)	7.9(1.9)
Identity	HED-strong	0.950(0.005)	0.959(0.005)	0.936(0.007)	0.957(0.007)	0.992(0.002)	0.945(0.003)	0.872(0.005)	0.967(0.003)	0.920(0.005)	8.1(2.8)
Style	HSV-strong	0.953(0.004)	0.956(0.004)	0.940(0.003)	0.959(0.007)	0.986(0.004)	0.932(0.005)	0.878(0.003)	0.976(0.001)	0.917(0.004)	9.0(2.6)
Style	HSV-light	0.940(0.011)	0.960(0.004)	0.944(0.007)	0.926(0.012)	0.992(0.001)	0.958(0.002)	0.852(0.008)	0.974(0.001)	0.921(0.003)	9.4(3.6)
Style	HED-strong	0.955(0.002)	0.949(0.004)	0.936(0.003)	0.954(0.005)	0.982(0.005)	0.942(0.002)	0.884(0.004)	0.975(0.000)	0.925(0.003)	9.9(1.2)
Grayscale	BC	0.956(0.003)	0.962(0.003)	0.935(0.005)	0.961(0.002)	0.989(0.002)	0.939(0.004)	0.851(0.002)	0.972(0.000)	0.884(0.003)	12.2(1.2)
Deconvolution	HSV-strong	0.955(0.003)	0.936(0.008)	0.941(0.004)	0.943(0.009)	0.991(0.001)	0.865(0.010)	0.867(0.004)	0.961(0.002)	0.928(0.001)	13.9(1.9)
LUT	HED-strong	0.934(0.006)	0.941(0.006)	0.925(0.006)	0.963(0.005)	0.989(0.002)	0.945(0.002)	0.871(0.005)	0.956(0.002)	0.945(0.001)	14.0(2.2)
Deconvolution	HED-strong	0.942(0.003)	0.962(0.003)	0.897(0.006)	0.967(0.003)	0.993(0.002)	0.827(0.018)	0.853(0.006)	0.969(0.001)	0.927(0.002)	14.4(1.2)
LUT	HSV-strong	0.923(0.009)	0.939(0.003)	0.928(0.005)	0.947(0.008)	0.987(0.002)	0.949(0.003)	0.862(0.007)	0.962(0.002)	0.940(0.002)	17.0(2.0)
Network	BC	0.944(0.003)	0.950(0.003)	0.903(0.003)	0.934(0.006)	0.983(0.005)	0.953(0.003)	0.869(0.009)	0.981(0.001)	0.881(0.005)	17.4(1.5)
Identity	HSV-light	0.888(0.013)	0.951(0.009)	0.942(0.004)	0.930(0.023)	0.962(0.015)	0.949(0.001)	0.905(0.005)	0.976(0.000)	0.894(0.003)	17.4(2.9)
LUT	HED-light	0.914(0.011)	0.926(0.011)	0.923(0.006)	0.932(0.019)	0.993(0.001)	0.948(0.003)	0.852(0.021)	0.966(0.003)	0.940(0.002)	17.9(2.2)
LUT	HSV-light	0.894(0.006)	0.936(0.006)	0.921(0.003)	0.942(0.007)	0.987(0.002)	0.951(0.002)	0.860(0.010)	0.971(0.002)	0.945(0.002)	19.2(1.2)
LUT	BC	0.925(0.025)	0.948(0.027)	0.853(0.016)	0.790(0.061)	0.985(0.004)	0.951(0.004)	0.848(0.018)	0.973(0.003)	0.924(0.005)	21.3(3.3)
Style	BC	0.949(0.005)	0.858(0.031)	0.938(0.001)	0.411(0.065)	0.987(0.004)	0.949(0.006)	0.764(0.047)	0.946(0.002)	0.903(0.005)	23.3(2.2)
Deconvolution	HSV-light	0.942(0.004)	0.930(0.009)	0.913(0.023)	0.961(0.002)	0.982(0.005)	0.850(0.019)	0.840(0.009)	0.958(0.006)	0.917(0.002)	23.5(1.0)
Network	Basic	0.944(0.003)	0.954(0.007)	0.887(0.010)	0.959(0.004)	0.969(0.005)	0.905(0.006)	0.815(0.019)	0.977(0.002)	0.855(0.006)	23.6(1.4)
Network	Morphology	0.939(0.010)	0.949(0.006)	0.890(0.012)	0.950(0.009)	0.980(0.006)	0.913(0.011)	0.823(0.022)	0.977(0.001)	0.868(0.002)	23.9(1.1)
Deconvolution	HED-light	0.930(0.005)	0.912(0.015)	0.916(0.005)	0.948(0.006)	0.982(0.002)	0.816(0.011)	0.834(0.004)	0.970(0.003)	0.927(0.005)	25.4(2.3)
Deconvolution	Morphology	0.951(0.003)	0.938(0.006)	0.849(0.021)	0.951(0.008)	0.993(0.002)	0.754(0.027)	0.749(0.037)	0.903(0.008)	0.865(0.015)	27.7(0.6)
Grayscale	Morphology	0.943(0.010)	0.820(0.021)	0.922(0.005)	0.941(0.011)	0.991(0.006)	0.910(0.009)	0.816(0.005)	0.929(0.006)	0.813(0.009)	27.7(1.2)
Style	Morphology	0.935(0.011)	0.725(0.082)	0.934(0.002)	0.361(0.113)	0.992(0.004)	0.918(0.006)	0.754(0.006)	0.873(0.010)	0.890(0.006)	28.6(2.4)
Grayscale	Basic	0.940(0.007)	0.692(0.064)	0.926(0.010)	0.938(0.019)	0.992(0.001)	0.882(0.008)	0.661(0.039)	0.934(0.002)	0.798(0.006)	30.0(0.6)
Deconvolution	BC	0.942(0.004)	0.896(0.008)	0.682(0.044)	0.949(0.005)	0.989(0.006)	0.794(0.021)	0.792(0.028)	0.930(0.007)	0.872(0.004)	30.4(1.2)
LUT	Morphology	0.898(0.007)	0.920(0.007)	0.801(0.021)	0.874(0.025)	0.969(0.008)	0.895(0.013)	0.803(0.007)	0.939(0.006)	0.906(0.006)	32.6(1.4)
Deconvolution	Basic	0.919(0.015)	0.896(0.038)	0.810(0.081)	0.902(0.026)	0.993(0.001)	0.753(0.006)	0.791(0.009)	0.903(0.003)	0.836(0.008)	32.8(0.7)
Style	Basic	0.918(0.002)	0.334(0.133)	0.926(0.004)	0.124(0.041)	0.991(0.003)	0.865(0.025)	0.723(0.024)	0.863(0.020)	0.857(0.010)	33.6(1.0)
LUT	Basic	0.908(0.010)	0.894(0.030)	0.809(0.022)	0.772(0.072)	0.951(0.009)	0.906(0.011)	0.741(0.018)	0.930(0.014)	0.890(0.013)	34.6(0.8)
Identity	BC	0.899(0.006)	0.634(0.100)	0.741(0.016)	0.177(0.047)	0.906(0.034)	0.936(0.006)	0.704(0.060)	0.684(0.009)	0.761(0.012)	36.2(0.4)
Identity	Morphology	0.811(0.026)	0.671(0.099)	0.673(0.027)	0.214(0.174)	0.986(0.006)	0.374(0.191)	0.602(0.023)	0.569(0.028)	0.720(0.009)	37.2(0.7)
Identity	Basic	0.811(0.009)	0.563(0.309)	0.790(0.047)	0.406(0.375)	0.965(0.009)	0.631(0.178)	0.624(0.053)	0.556(0.057)	0.701(0.028)	37.6(0.5)

Tellez, D. et al. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Med. Image Anal.* **58**, 101544 (2019).

Diverse training data

- The natural variability in the training data matters.
- Artificial and natural data variation complement each other.

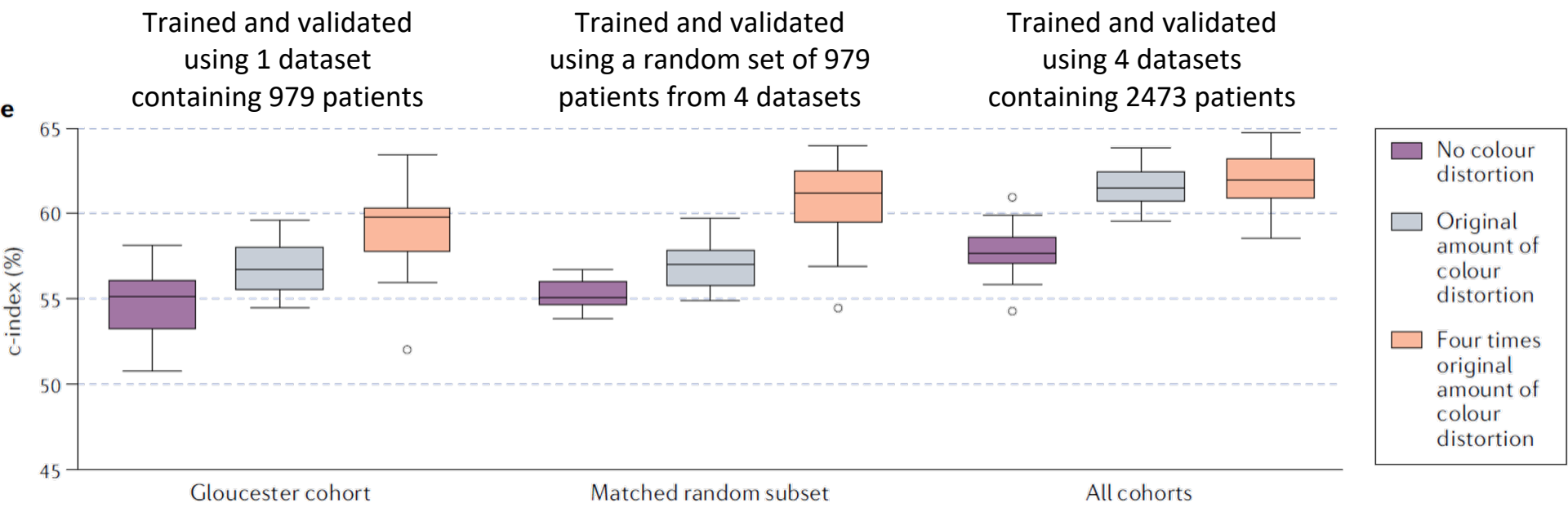


Figure 2e in Kleppe, A. et al. Designing deep learning studies in cancer diagnostics. *Nat. Rev. Cancer* **21**, 199–211 (2021).

Too much distortion

- Trade-off between facilitating the learning of relations that generalise well beyond the development dataset and not occluding relevant information for the prediction task.

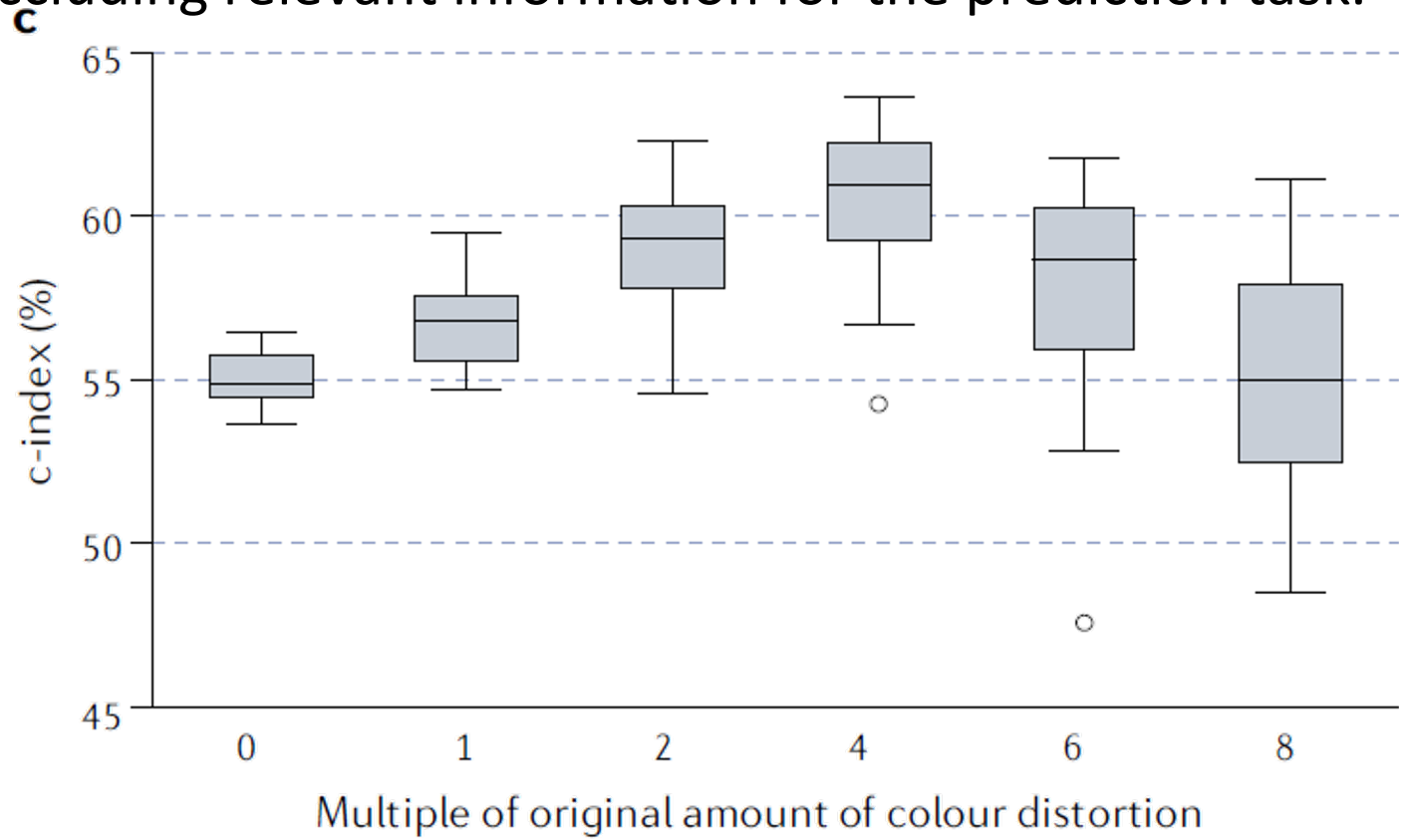


Figure 2c in Kleppe, A. et al. Designing deep learning studies in cancer diagnostics. *Nat. Rev. Cancer* **21**, 199–211 (2021).

Progress

- Internal testing (recap)
- External testing
- Facilitating generalisation
- **Some performance metrics**
- Estimator uncertainty

Performance metric

- Performance metric is often chosen by convention.
- While this should be reported, a model envisioned for practical usage should primarily be evaluated using a performance metric that reflects the intended application.
  - If the end user of your model will simply use the classifications, then the performance metric should also only use the classifications (not the continuous prediction scores).
    - Note that metrics calculated from classifications are often more conservative than metrics calculated from prediction scores.
  - If this is not done, then it can be difficult to interpret how well the model will perform in the intended application, both in itself and relative to other models.
- Some performance metrics ignore relevant aspects of an evaluation and should be complemented by other statistics.

Confusion matrix

- Each element specifies the number of samples with a particular class c_A predicted to be a particular class c_P

		Predicted class			
		$\mathbf{c_1}$	$\mathbf{c_2}$	…	$\mathbf{c_k}$
Actual class	$\mathbf{c_1}$	$n_{1,1}$	$n_{1,2}$	…	$n_{1,k}$
	$\mathbf{c_2}$	$n_{2,1}$	$n_{2,2}$	…	$n_{2,k}$
	…	…	…	…	…
	$\mathbf{c_k}$	$n_{k,1}$	$n_{k,2}$	…	$n_{k,k}$

c_i specifies a class and $n_{a,b}$ specifies a count

- Diagonal elements specify correctly classified samples.

Confusion matrix

- In the special case of two classes, the confusion matrix contain only four values, often named as:

		Predicted class	
		Predicted positive	**Predicted negative**
Actual class	**(Actual) positive**	True positive (TP)	False negative (FN)
	(Actual) negative	False positive (FP)	True negative (TN)

Useful in object detection

- Assume a model is used to detect objects of a single class “animal”:



Precision and recall

- Can be deduced from the confusion matrix for two classes.

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

Precision = Accuracy of positive predictions

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

Recall = Percentage of positive elements found

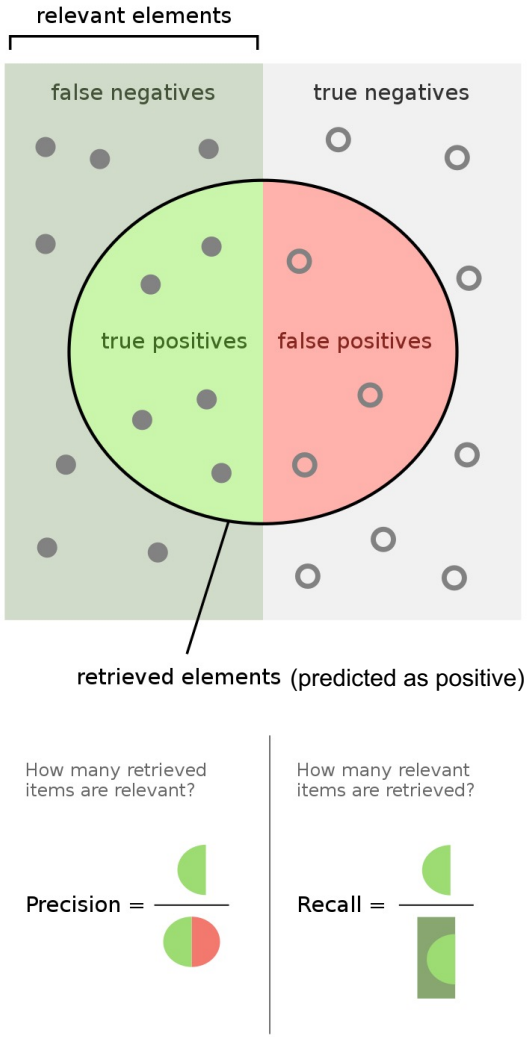
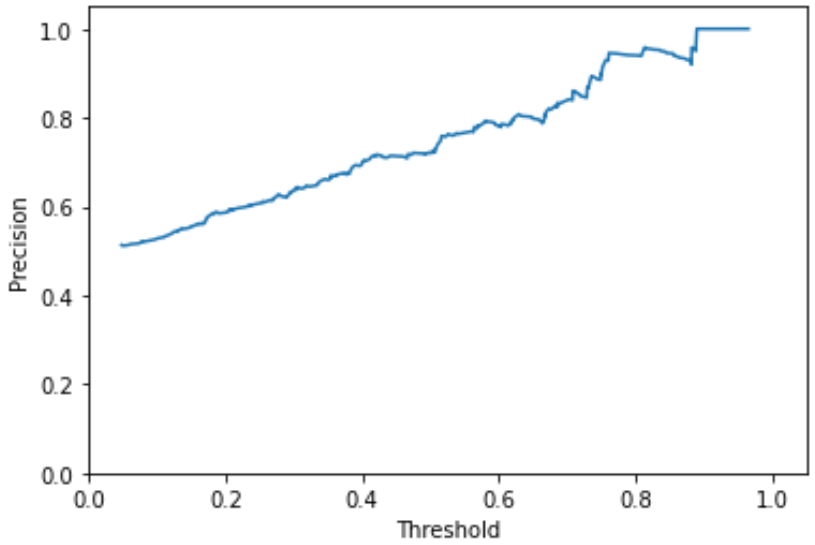
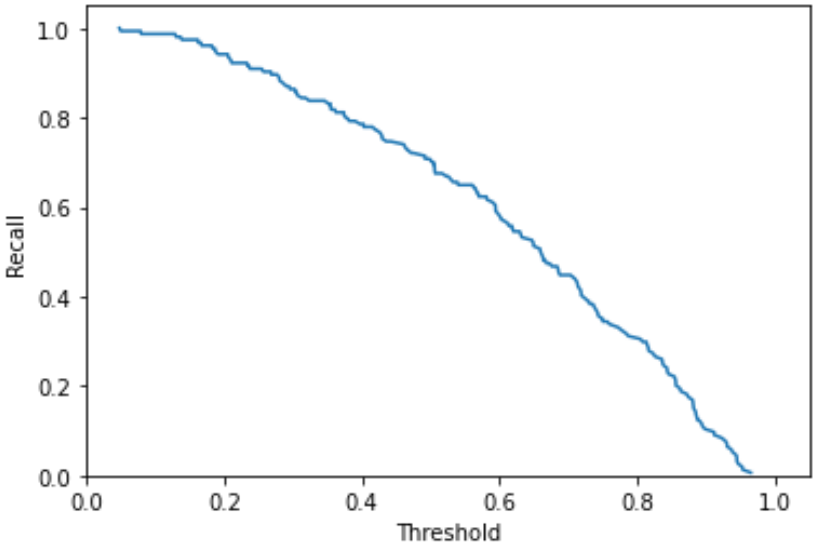


Image source: By Walber - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=36926283

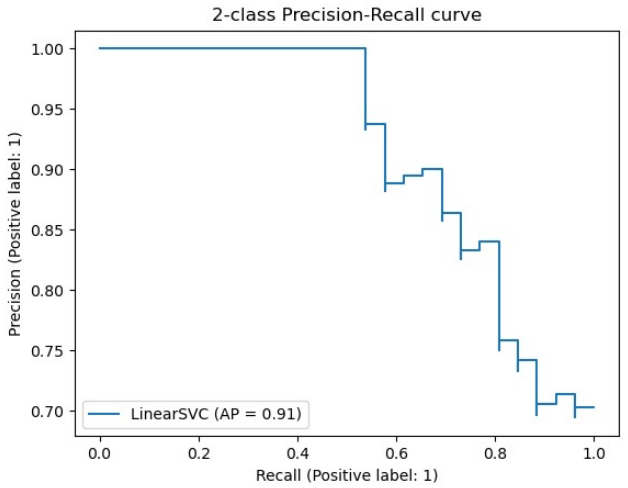
Precision and recall





Precision-recall curve

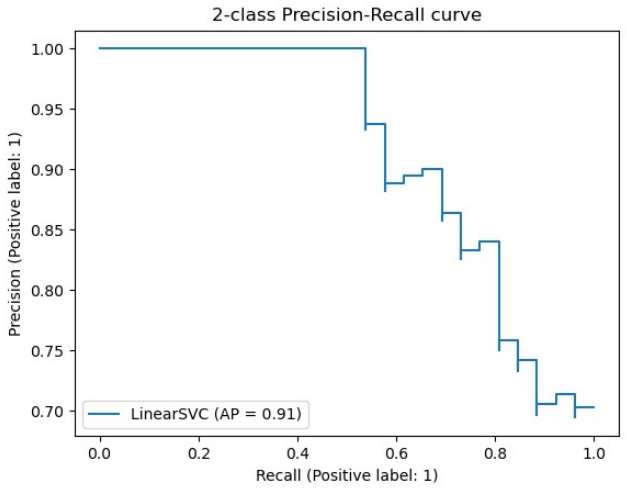
- A curve depicting the pairs of precision and recall for different thresholds of the prediction score of the positive class.



https://scikit-learn.org/stable/_images/sphx_glr_plot_precision_recall_001.png

Precision-recall curve

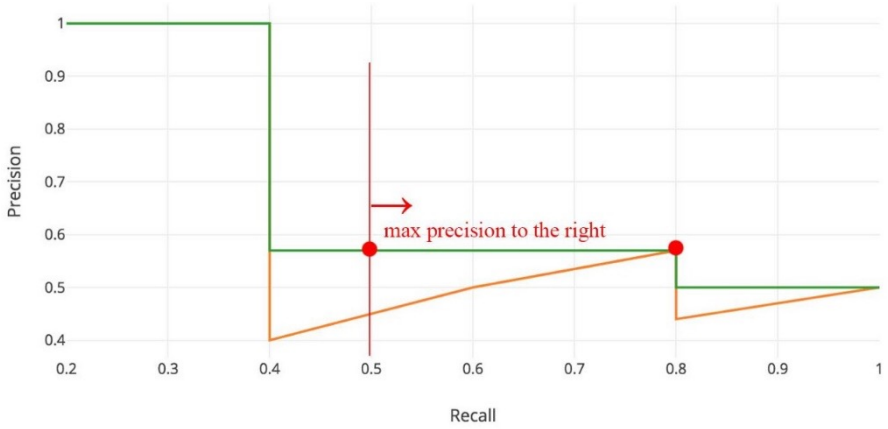
1. Compute positive prediction scores for each sample in the evaluation set.
2. Start with a threshold larger than all scores.
   - Gives no positive predictions, so recall is 0 and precision is undefined (the expression is 0/0, can define it as 1).
3. Repeat until the threshold is below all scores:
   a. Lower the threshold to include one more of the observed scores.
      - Note that multiple samples may have this score.
   b. Calculate the recall and precision when using the new threshold.
   c. Draw a curve between the new and the previous (recall, precision) pair.
      - There are multiple choices for how to do this.
      - In scikit-learn above 0.19:
        Draw a horizontal line between the new and previous recall at the new precision and then a vertical line between the new and previous precision at the previous recall.



https://scikit-learn.org/stable/_images/sphx_glr_plot_precision_recall_001.png

Average precision (AP)

- The area under the precision-recall curve is called the *average precision* (for the positive class).
- Measures how well the prediction scores match the class labels.
  - Higher is better; the AP of a perfect classifier is 1.
  - AP for a random classifier is equal to the proportion of actual positives in the evaluation set.
- Some compute the area under an interpolated version of the precision-recall curve, but the "wiggles" this addresses depends on how a curve is drawn between adjacent (recall, precision) pairs.
  - Such "wiggles" are not a prominent issue when drawing curves between pairs as in scikit-learn above 0.19.



https://miro.medium.com/max/1400/1*pmSxeb4EfdGnzT6Xa68GEQ.jpeg

Average precision (AP)

Alternative approach to compute AP:

1. Compute positive prediction scores for each sample in the evaluation set.
2. Sort samples by their scores in descending order (i.e., sample with highest score first).
   - Secondary sort on ascending label, implying that if multiple samples have the same score, then actual negative samples should come before actual positive samples with the same score.
3. Calculate:

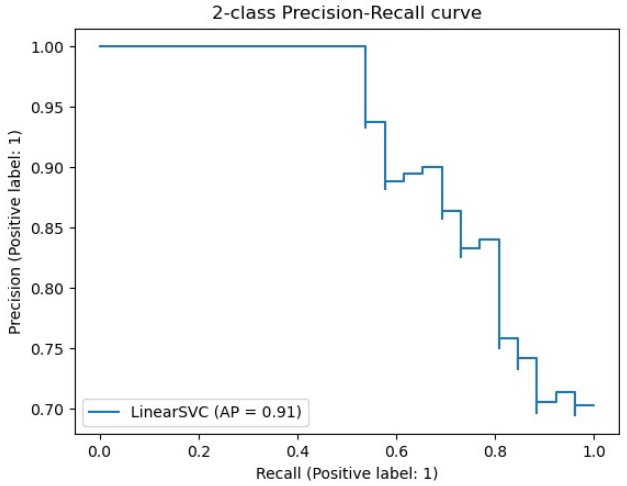
$$AP = \frac{1}{R}\sum_{k=1}^{N} y_k P@k$$

- R is the number of actual positives in the evaluation set:

$$R = \sum_{k=1}^{N} y_k = TP + FN$$

- N is the number of samples in the evaluation set.
- y_k is 1 if sample k (i.e., sample with index k after sorting) is an actual positive, otherwise 0.
- $P@k$ is precision at k, i.e., the precision for the first k samples after sorting:

$$P@k = \frac{1}{k}\sum_{i=1}^{k} y_i$$



https://scikit-learn.org/stable/_images/sphx_glr_plot_precision_recall_001.png

Mean average precision (mAP)

- If there are more than two classes,
  then the average precision can be calculated per class.
  - "Positive" is the relevant class and "Negative" is all other classes.
  - Perform the calculation for all classes.
- The arithmetic mean of the average precisions is the mAP:

$$\mathrm{mAP} = \frac{1}{k}\sum_{i=1}^{k} AP_i$$

where *k* is the number of classes

- Arithmetic means of other sets of average precisions are also referred to as mAP.

Sensitivity and specificity

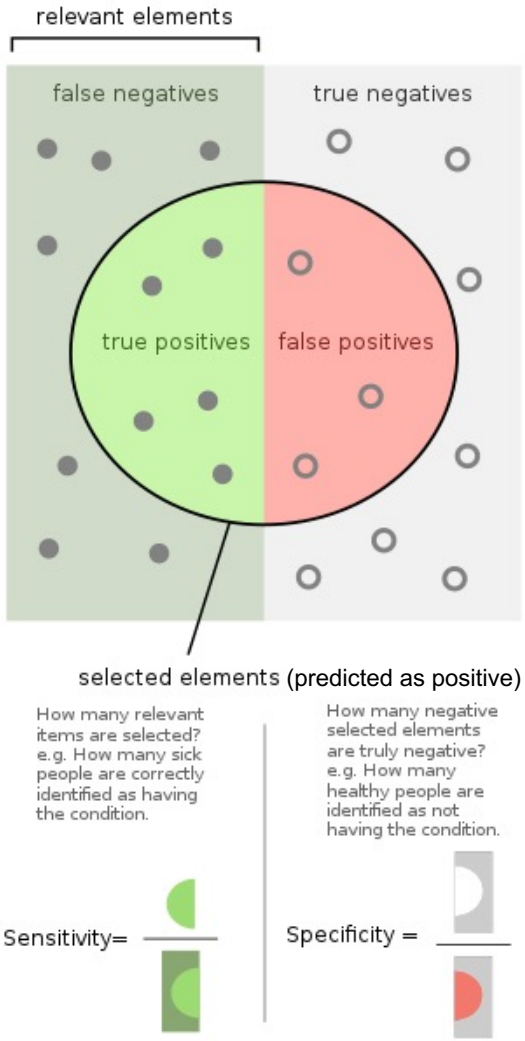
Can be deduced from the confusion matrix for two classes.

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

- Proportion of actual positives that are positive predictions.
- Same as recall (a.k.a. true positive rate).

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

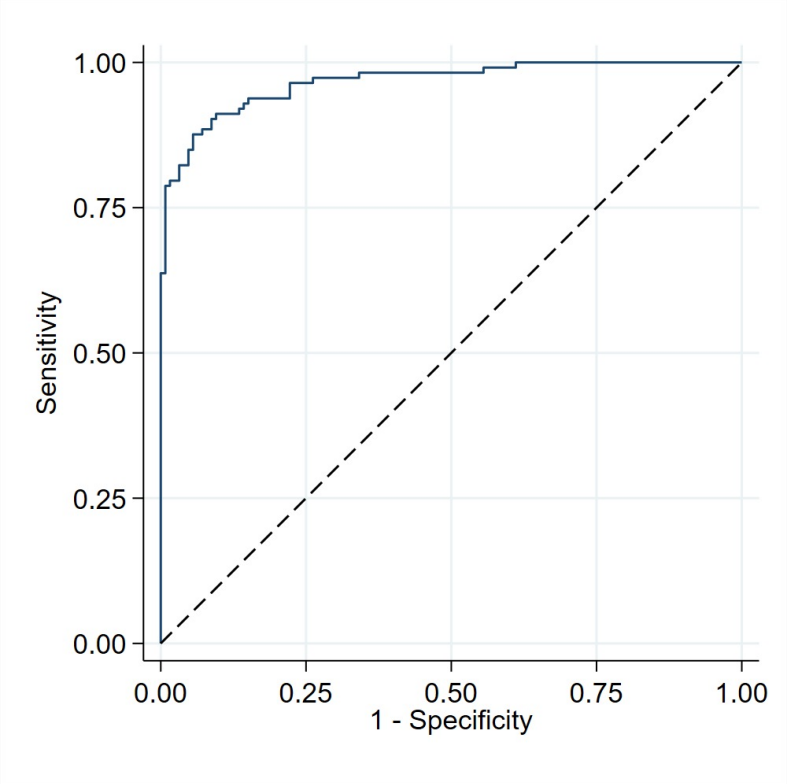
- Proportion of actual negatives that are negative predictions.
- A.k.a. true negative rate.
- False positive rate = 1 - Specificity



Receiver Operator Characteristic (ROC) curves

ROC curves can be created similarly as precision-recall curves, except that:

- Using each threshold, the sensitivity and specificity is calculated and plotted instead of the recall and precision.
- Linear interpolation is used between the new and the previous (1-Specificity, Sensitivity) pair.
- ROC curves start at (0,0) and end at (1,1).



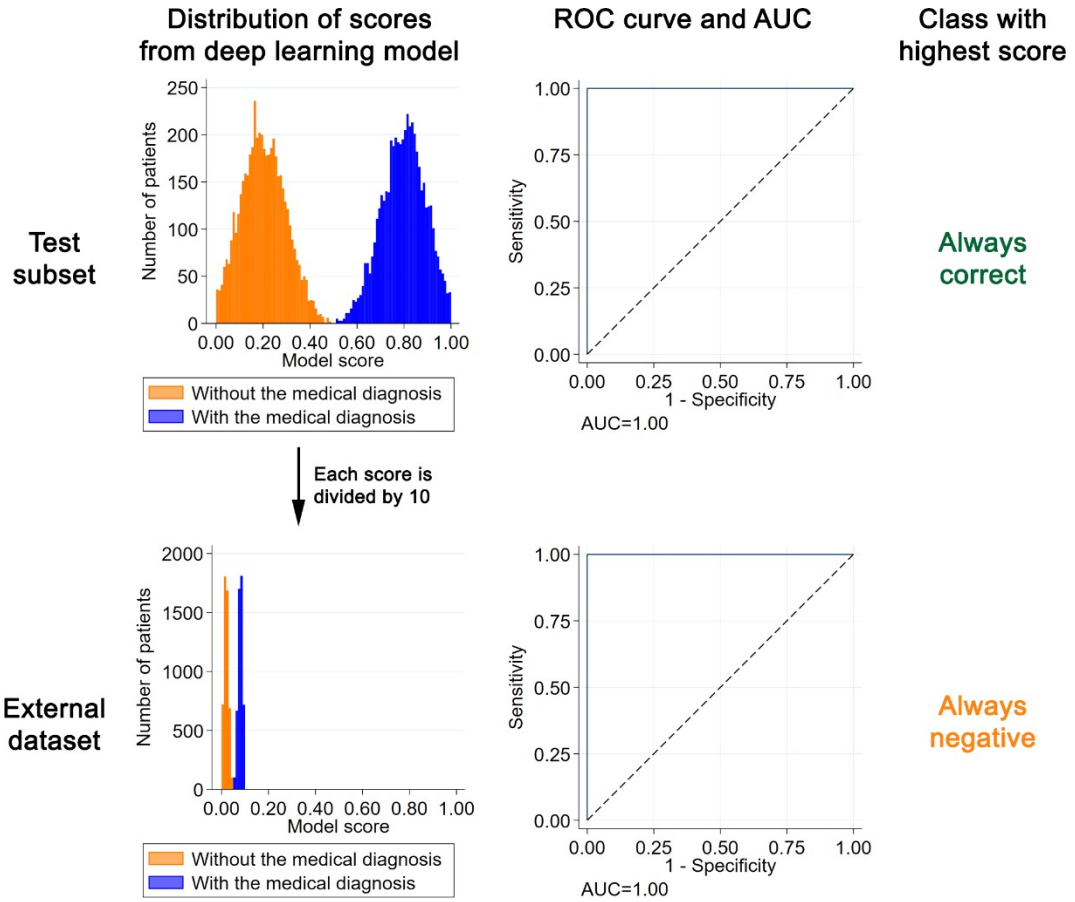
Area under the curve (AUC)

- AUC and AUROC is the area under the ROC curve.
- Create all possible pairs with one element chosen from the positive class and one from the negative class. It can be shown that AUROC is the proportion of pairs where the prediction score of the positive class > prediction score of the negative class.
  - If the scores are identical, then 0.5 is added (instead of 1 or 0) to the nominator of the proportion being calculated.
- Thus, the AUROC can be viewed as the probability that the predicted score for the positive class is larger for an actual positive sample than for an actual negative sample.

AP and AUROC

- Summarise the discriminatory value of the prediction scores.
- Only use the ranking of the prediction scores,
  not their absolute values.
- Are thus invariant to any strictly increasing transform.
  - That is, the metric value of the transformed prediction scores will be the same as the metric value of the original prediction scores.
- A concern when combined with external testing.
  - A distribution shift between the development dataset and the external dataset can result in changes of the score distribution that resemble those caused by some strictly increasing transform.
- AP and AUROC may thus
  hide a type of model bias that is not uncommon.

AP and AUROC



- What if the scores of a second external dataset was transformed by $s^{1/10}$ compared to the test subset?
  - AUC will still be 1.00, but the class with highest score will nearly always be positive (specifically if $s > 0.5^{10} \approx 0.00098$).

AP and AUROC

- To overcome the shortcomings of AP and AUROC, one could use any other metric (like balanced accuracy) along with AP/AUROC.
- The other metric could reveal the any biases and shortcomings that AP or AUROC hide.

Balanced Accuracy

		Predicted class	
		Predicted positive	**Predicted negative**
Actual class	**(Actual) positive**	True positive (TP)	False negative (FN)
	(Actual) negative	False positive (FP)	True negative (TN)

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Recall / Sensitivity / True Positive Rate (TPR) = $\frac{TP}{TP+FN}$
- Specificity / True Negative Rate (TNR) = $\frac{TN}{TN+FP}$
- Balanced Accuracy = $\frac{TPR+TNR}{2}$

- Balanced accuracy is useful when classes are imbalanced

Balanced Accuracy

Actual class \ Predicted class	c_1	c_2	…	c_k
c_1	$n_{1,1}$	$n_{1,2}$	…	$n_{1,k}$
c_2	$n_{2,1}$	$n_{2,2}$	…	$n_{2,k}$
…	…	…	…	…
c_k	$n_{k,1}$	$n_{k,2}$	…	$n_{k,k}$

c_i specifies a class and $n_{a,b}$ specifies a count

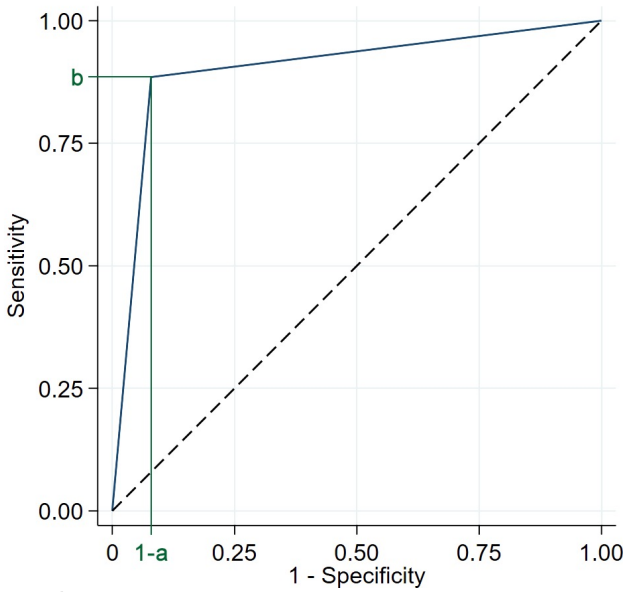
- Accuracy = $\frac{n_{11}+n_{22}+\ldots+n_{kk}}{(n_{11}+n_{12}+\ldots+n_{1k})+(n_{21}+n_{22}+\ldots+n_{2k})+\ldots+(n_{k1}+n_{k2}+\ldots+n_{kk})}$
- Recall / Sensitivity of c_1 = S_{c1} = $\frac{n_{11}}{n_{11}+n_{12}+\ldots+n_{1k}}$
- Similarly calculate sensitivity of all the classes
- Balanced Accuracy = $\frac{S_{c1}+S_{c2}+\ldots+S_{ck}}{k}$

Performance metric

- For many classification tasks, the ultimate goal is to simply predict the correct class.
- If so, a classification metric such as balanced accuracy may reflect the intended application.
  - If defining the sensitivity of a particular class as the proportion of samples with that actual class that are predicted to be that class, then the balanced accuracy is the average of all classes' sensitivities.
- Any metric calculated using classifications rather than prediction scores will give lower performance estimate in the case of the aforementioned type of model bias.
  - Assuming the way that classifications are obtained from prediction scores is not influenced by the external dataset.
    - This should therefore be pre-specified, e.g., as "select the class with largest prediction score" or "apply these thresholds" (which might be thresholds identified as suitable using the development dataset).

Performance metric

- Also, the AUROC calculated from classifications instead of prediction scores is a reasonable metric.
- It is actually equivalent to the balanced accuracy in the case of two classes:



  - Let b be the sensitivity and a the specificity of a dichotomous classifier (see figure above).
  - The AUROC is the sum of the area of one rectangle and two triangles.
  - $$AUC = \big(1-(1-a)\big)b + \frac{(1-a)b}{2} + \frac{\big(1-(1-a)\big)(1-b)}{2}$$
    $$= ab + \frac{b-ab}{2} + \frac{a-ab}{2} = \frac{a+b}{2}$$

Progress

- Internal testing (recap)
- External testing
- Facilitating generalisation
- Some performance metrics
- **Estimator uncertainty**

Uncertainty of performance estimate

- When estimating the performance using a finite set of data, the estimate is always associated with an uncertainty.
  - Technically, the estimator (i.e., the rule for calculating the estimate) has a distribution around the true performance.
- This uncertainty should be quantified and reported with the estimate of the performance metric.
- Common to estimate a 95% confidence interval.
  - A procedure for creating 95% confidence intervals for a parameter (e.g., a performance metric) should be such that on average 95% of created confidence intervals contain the true value of the parameter.
  - Often misinterpreted as “95% probability of containing the true value”. But given a concrete confidence interval, the interval either contains the true value or not (it is no longer a matter of probability).

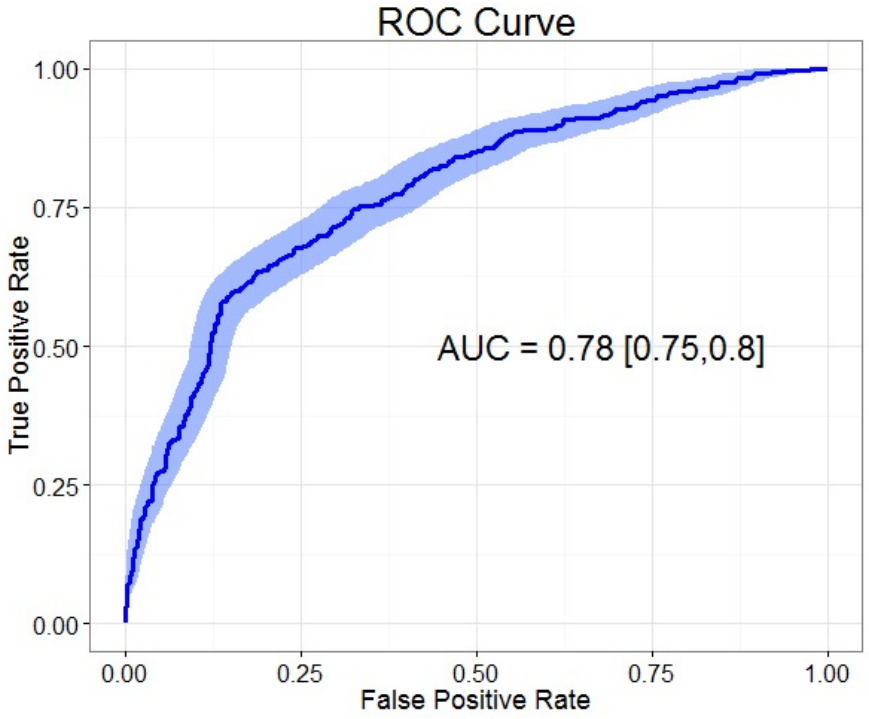
Uncertainty of performance estimate

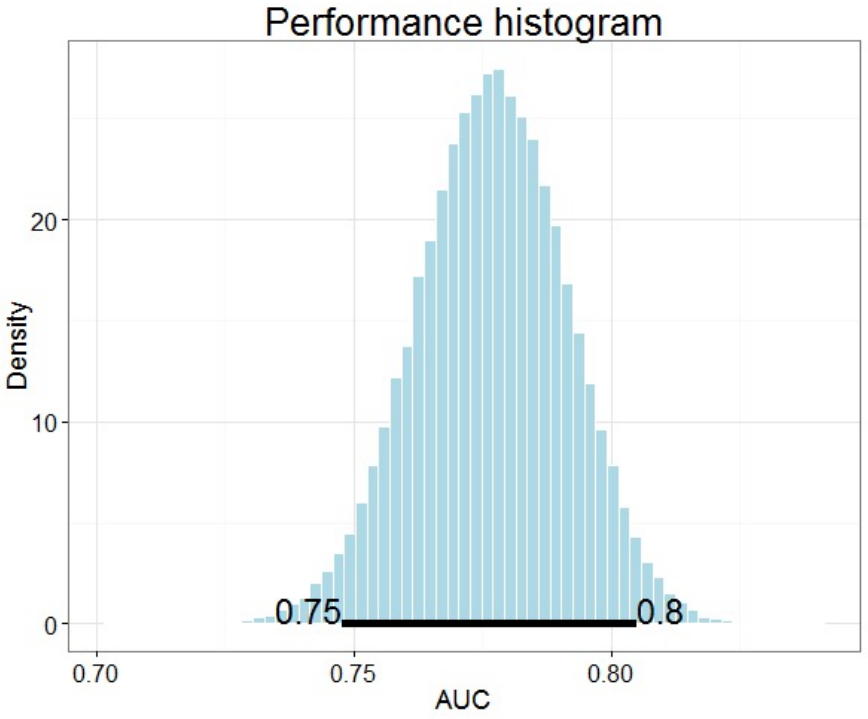
- For many performance metrics, there exists specific procedures for estimating confidence intervals.
  - E.g. for sensitivities, one of the formulas for binomial proportion confidence interval could be applied, e.g. Clopper-Pearson interval or normal approximation interval.
- Bootstrap confidence intervals are common and can be calculated for (almost?) all metrics.
  - Although bootstrap confidence intervals might be a bit narrow because of the similarity between the sampled sets.
    - Isaksson, A., Wallman, M., Göransson, H. & Gustafsson, M. G. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognit. Lett.* **29**, 1960–1965 (2008).

Bootstrap confidence interval

Create a distribution of the parameter by repeating e.g., 10000 times:

1. Sample with replacement from the original test set to obtain a resampled test set with the same size as the original test set.
2. Compute the parameter of interest using the resampled test set.





https://i2.wp.com/www.epeter-stats.de/wp-content/uploads/2015/05/example-1.png

Bootstrap confidence interval

- The standard bootstrap confidence interval is computed by estimating the standard error of the bootstrapped distribution of the parameter, *s*, and constructing a confidence interval when assuming normal distribution:
  - (<point estimate of parameter> - $z_{\alpha/2}s$, <point estimate of parameter> + $z_{\alpha/2}s$)
  - $z_{\alpha/2}$ is the $100(1-\alpha/2)^{th}$ percentile of the standard normal distribution, e.g., about 1.96 if constructing a 95% confidence interval ($\alpha=0.05$).

- There exists other bootstrap confidence intervals that are based on the percentiles of the bootstrapped distribution of the parameter.
  - These are better in particular if the performance estimate (i.e., the point estimate of the parameter) is close to its theoretical maximum, e.g., 100%.
  - Efron, B. Better Bootstrap Confidence Intervals. *J. Am. Stat. Assoc.* **82**, 171-185 (1987).

Uncertainty of performance estimate

If the model performs overoptimistically on the test set, then the confidence interval will inherit this overoptimism.

- This may also happen for external datasets that are used for model selection through testing multiple options and selecting the best one.

Key learning goals

- Be able to design projects in a way that makes the performance estimation as reliable as is practically feasible.
- Be able to critically reason about the reliability of evaluations of the performance of a trained neural network model.
- Be able to suggest approaches for facilitating generalisation, in particular generalisation to external data.
- Understand AP and AUROC and limitations of performance metrics that only apply the ranking of prediction scores.
- Interpret a performance estimate as a value with associated uncertainty, and be able to estimate this uncertainty.