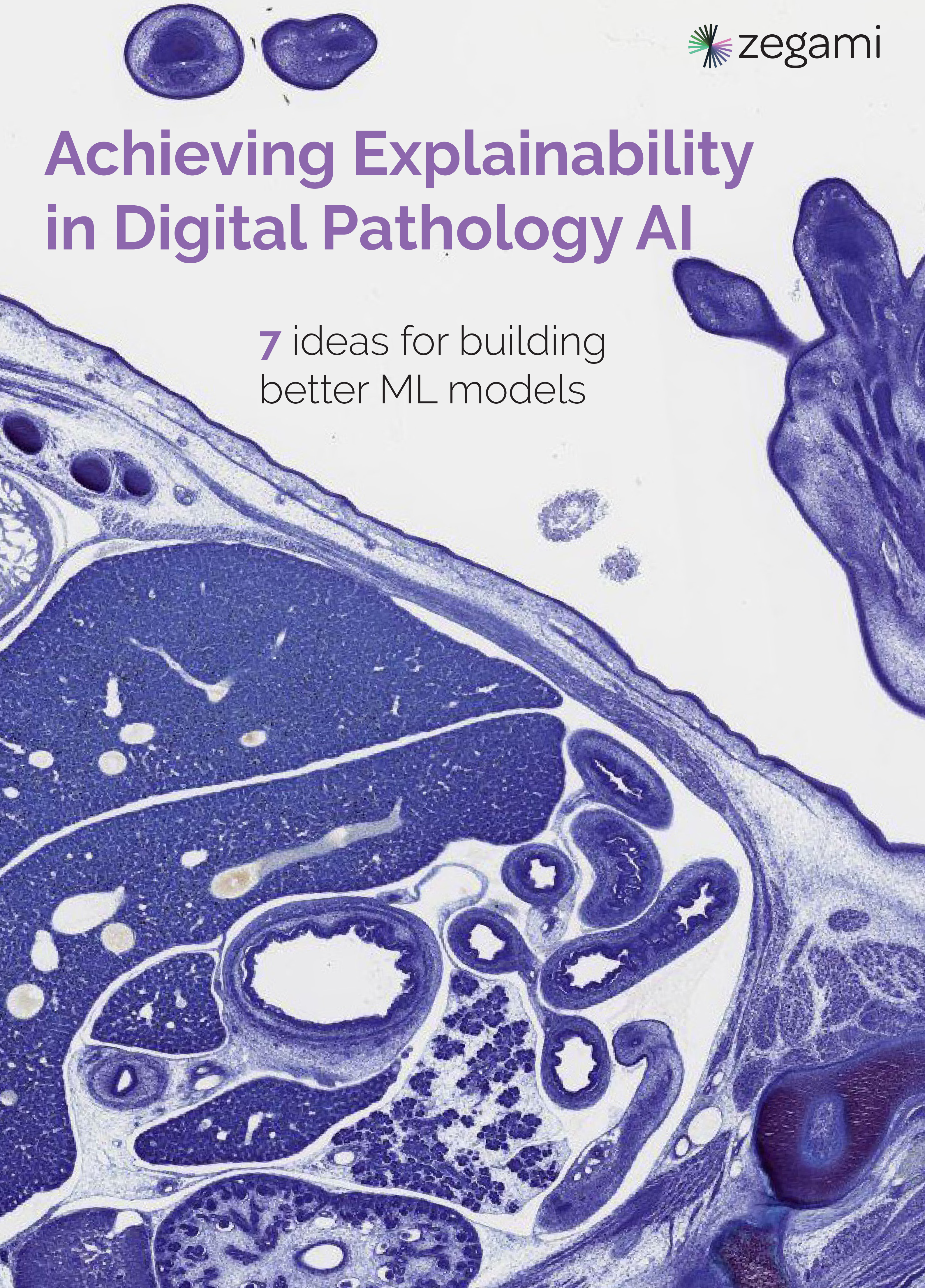
zegami

Achieving Explainability in Digital Pathology AI

7 ideas for building better ML models



Achieving Explainability

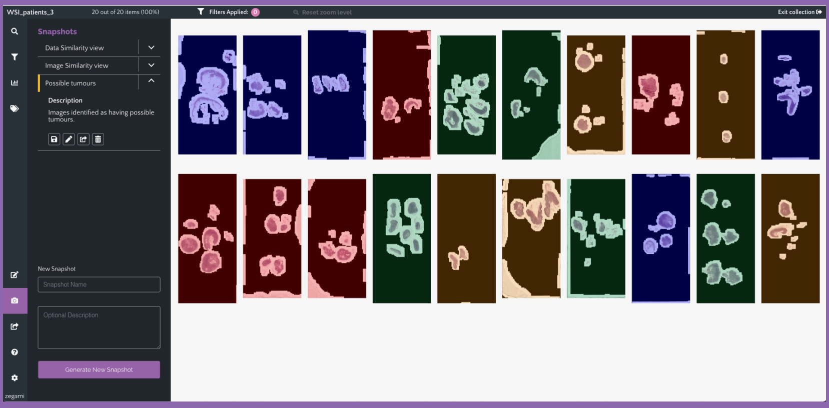
Building Digital Pathology AI models is hard and it can take a lot of trial and error to get them working, and prove that they are robust.

We typically work with image detection and object classification models, but you may be able to extrapolate some of these points to other domains.

We've put together this list of tips, based on our experience, to help you build more accurate, less biased models.

Digital pathology makes it easier to store, view and organise medical images, making them easier to access and share. It also facilitates rapid engagement and collaboration (for example in multi-centre studies) between professionals. Second opinions can be easily and quickly solicited from clinicians in other locations, saving time and costs, and reducing the risks associated with physically transporting slides. Furthermore, organising images in a transparent and consistent way makes it easier for clinicians and scientists to satisfy regulatory requirements.

Crucially, the increasing prominence of digital pathology has provided access to specialist software applications such as automated image analysis tools. These are used to assist in the interpretation and quantification of medical images. These tools employ sophisticated algorithms – often based on data from thousands of subjects. Computational algorithms, when correctly applied, are objective, highly accurate and can provide pathologists with more information to help make a diagnosis.



More recently this means using artificial intelligence (AI) for image analysis. There are many examples of the use of AI or computer vision for analysing images in scientific research. For example, the ability to detect, count and classify objects in images is a very important and challenging task. It is used in many areas, including image processing, medicine and other sciences. For example, it is used to analyse satellite images to count trees, cars or people in a city. It is also used to identify tumour cells in x-rays to detect cancer.

In digital pathology, getting enough clinical samples is a starting point, but even then building digital pathology AI models is hard. How do you know if you have created a decent AI model? The answer is simple: you don't. You're almost certainly going to have to go through a fair bit of trial and error to get the right network architecture, hyper-parameters and learning rate. .

Zegami users work with a variety of image detection and object classification models. The principles we explain here are applicable across the board to imaging including digital pathology. To achieve this, it is necessary to assess a model's quality, which is why we have developed our 7 steps to explainability.

1 Understand what bias is

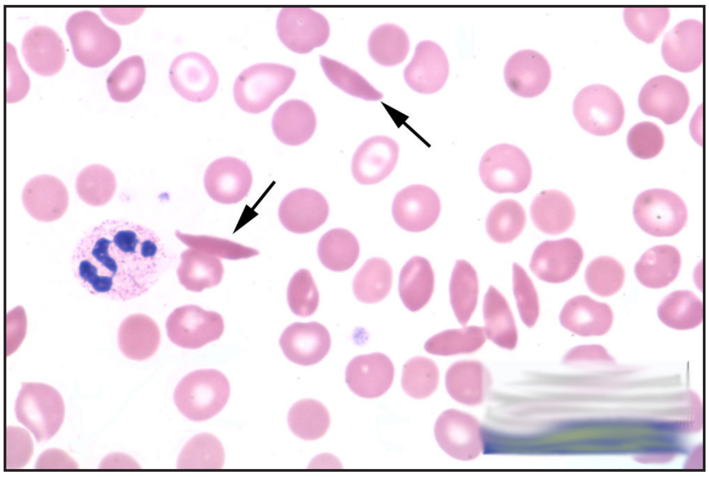
Bias can be the silent antagonist to training any model. You have made sure to gather a large hoard of input data, you've split it into your correct subsets with a generous portion of validation images to test with. You're also well aware of *overfitting*, diligently looking out for signs of it in your training/validation loss outputs (and hopefully applying some *data augmentation* too).

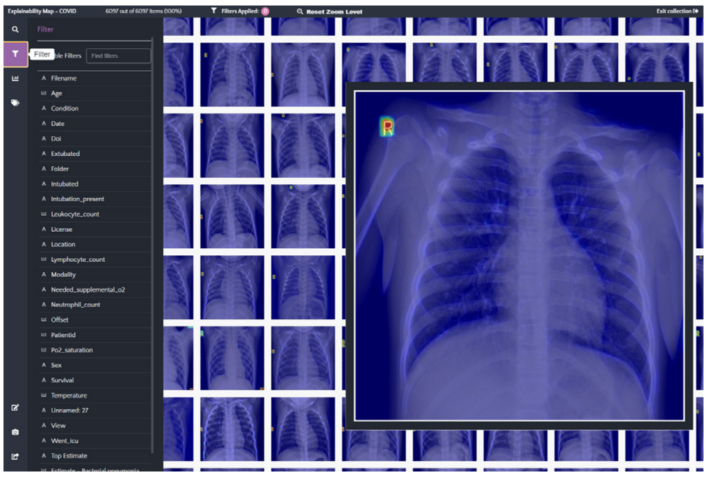
After pouring some time into training, your training accuracy is great. Your validation accuracy is great. It's exciting, you've trained a model that can intuitively make a prediction on the portion of your dataset that it has never seen before. Job done!

Unfortunately, this is a trap many can fall into (myself included). Your data can contain unintended themes and giveaways that you may have missed, that your model is an expert at finding and leveraging. These patterns are what we call a bias, and can exist throughout your training, validation and testing sets, and as your model only cares about getting the highest score it can, it will use these themes to cheat.

As an example, let's say you're training an image classification model to predict whether a slide of blood cells exhibit the signs of sickle cell anemia. You have 5000 slides of regular red blood cells from a hospital database, and 5000 of sickle cell kindly donated by a lab specialising in the condition's research. It is unlikely that the lab researching this condition is using the exact same procedure, microscopes, and slide annotation protocol as the regular cell slides you got from a hospital's database. Maybe the lab puts the date in the corner, or the hospital places the patient ID on the side. Maybe the lab uses a brighter backlight for their slides. Whatever the differences, if a difference can be at least somewhat reliably associated to a class, you have a bias, and your model may completely ignore looking at the cells when it can accurately predict (on the training, validation and testing sets) using these cues.

This happens a great deal. I have run into this issue with a hospital that donated healthy patient images with an 'R' placed on their right side. It seemed innocuous but diving into some *explainability* really highlighted what an issue this was.


A common bias present in medical data – clinicians place arrows on slides to help indicate the problem. These are often saved into the image and act as input data.



A handful of the 'healthy' class in a dataset. After training the model, we can map where the model paid most attention to come to its conclusion. Notice the hot 'R's and the complete disregard for the important data. attention to come to its conclusion. Notice the hot 'R's and the complete disregard for the important data.

2 Understand your training data

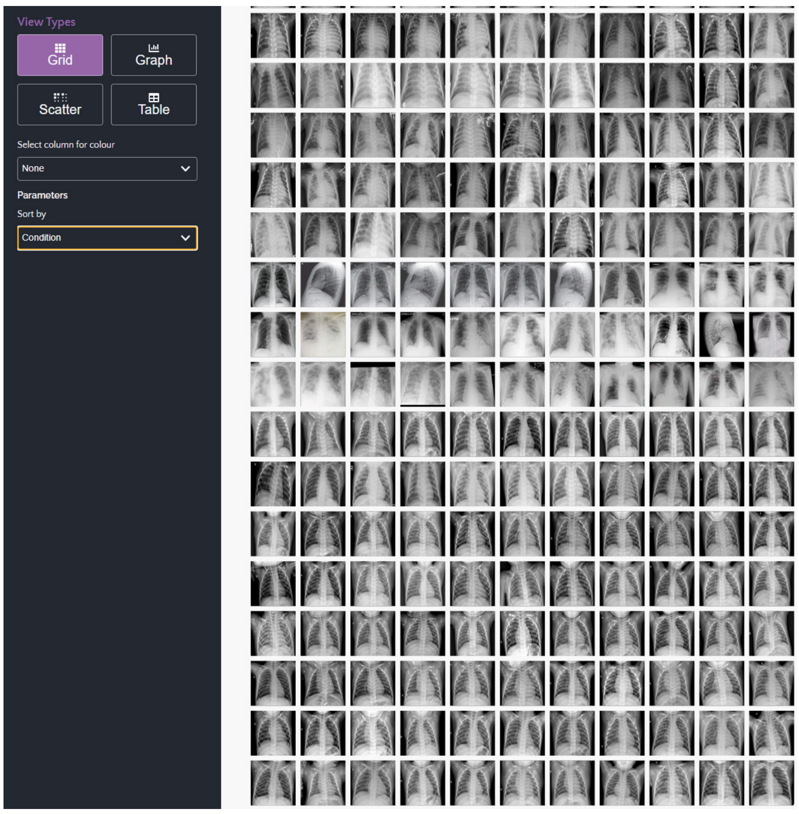
Part of the solution to the bias problem is simply understanding your data, in a broad scope. Check that the quality and distribution of input data is well balanced.

An example of this could be a cat/dog classifier with a training data set of 9000 images of cats and 1000 of dogs. Without addressing the imbalance here, the resulting model will be far more accurate with cats. Certain measures can be made during training to correct for slight imbalances, like setting penalty/reward weights higher for underrepresented classes. Data augmentation should always be used to bulk up your numbers too and can be used to tighten under/over-representation gaps.

If the differences between classes are subtle, separate all of them out next to each other and see if you can tell which group is which by looking at the low-resolution versions of the images. If you can see differences between the groups based on features not associated with the target object, your model will likely figure it out too.

Ultimately, this often overlooked first-pass using our natural intuition can save a great deal of time. Always remember that the model is trying to cheat, and it's your jot to try and stop it. Geography, machine calibration, demographics, language, ambient light, even putting an ID in the corner, can all ruin a dataset. Without delving into model explainability, these problems can be overlooked entirely.

You can spot bias simply by looking at your data. This view shows two classes of patients - one class came from multiple sources and the other from a single hospital. If our eyes can detect this bias, so can a model.



Select column for colour: None

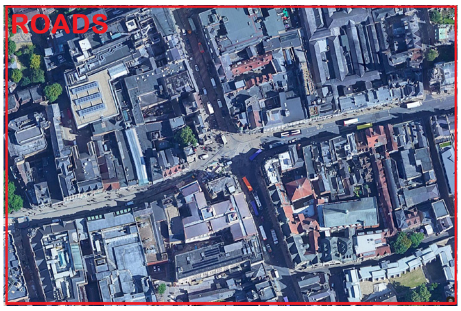
Parameters

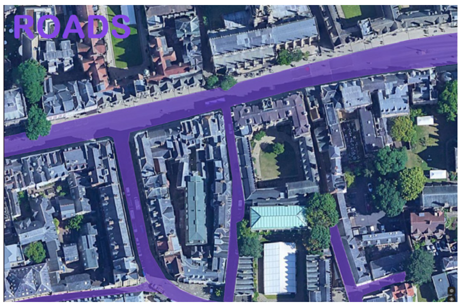
Sort by: Condition

3 Get better at annotating

Good annotation is key. Everyone has heard the garbage in, garbage out phrase when it comes to working with data, but it is worth reiterating. Feeding poor annotations into a model is analogous to tutoring a student mathematics, telling them that 4 x 4 = 17, and then being disappointed when they can't do maths properly.

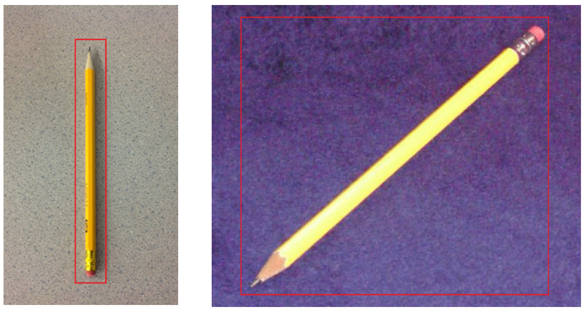
Assuming you're not providing wrong annotations, it's important to try and use the highest quality annotations that capture as much relevant information as possible. In the context of object detection (instances of a class in a picture), many architectures focus purely on bounding-box annotation. While this works fine, it doesn't tell the whole story, and performs poorly for some tasks.In the context of object detection (instances of a class in a picture), many early architectures focused heavily on bounding-box annotation. While this works fine, it doesn't tell the whole story, and less adequately for some tasks.



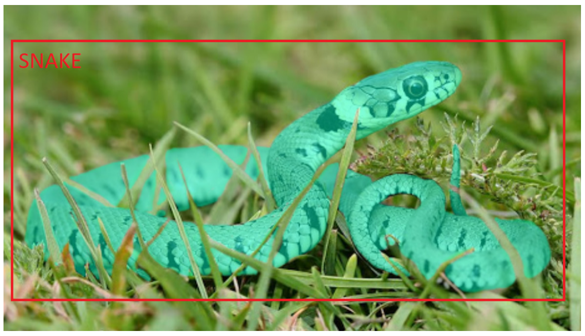


Imagine you were tasked with segmenting out the roads in a satellite view of a street. If you enclose the roads in a box, you're typically going to box the whole image. If your model is designed to mirror this input, you're not going to get much use from the results: For less extreme cases, using bounding boxes introduces needless error for all sorts of shapes. Compare the unneeded area used in these pencil labels by arbitrary orientation differences:

For less extreme cases, using bounding boxes introduces needless error for all sorts of shapes. Compare the unneeded area used in these pencil labels by arbitrary orientation differences:



Or a snake hiding amongst grass



In this example, the bounding box encloses 76% of the image, while the segmentation mask covers 19% of it. In machine learning we try to add enough bounded examples to cancel out the differences between samples to beat this issue – but that just brings us back to bias. Snakes aren't often photographed in the snow, and huskies are.

There are many cases where instance segmentation is a requirement (try separately labelling two differently colored wires twisted around each other with bounding boxes!), but probably most useful is the ability to gather more relevant information about an object. If your model can segment out objects in a microscopy slide it can calculate the circularity and area of an instance, and possibly extrapolate a sphericity and volume of an object.

All segmentation above was done using Zegami's own visual dataset creation tool.

4 Explore your results

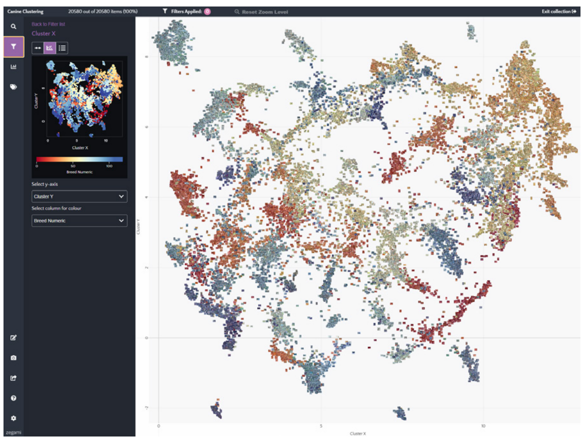
Just as it is crucial to understand your input data, scrutinising results should be under several lenses. A model's validation score is only an indicator of success, useful for detecting overfitting, but cannot be implicitly trusted for quality assurance. Mentioned above, the validation score will not tell you if a model is actually generalising to the desired concept, or just taking advantage of some unanticipated bias in the data. If there is one take-away from this section, it is this: *Always assume your model cheated.* It is a data-scientist's role to prove to the world this isn't the case - because all too often it is.

As proud humans who put in a lot of work to see their models succeed, we tend to be optimistic when inspecting the output of models. The output may not be perfect yet, but that can be solved with more data and more training time, right? While this generally helps, if you are going off a validation score and a glance over a few *well-chosen* inferences, it is all-too convenient to assumeany unexpected QA problems are simply the result of a lack of time/data.

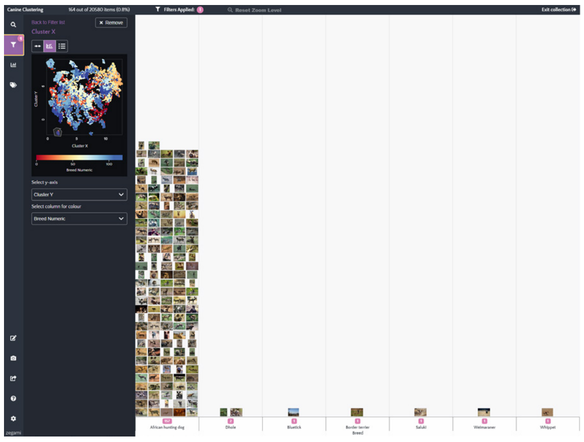
Inference all the test data you have, not just a few you think are "especially good test candidates". What you think is a good candidate may be trivial for your ultra-abstract model, it may even trip on the "easiest" example you have. Using Zegami, displaying tens of thousands of images at once and exploring confidence scores is one way to go. To really get an intuitive sense of how your model is seeing your data though we can dive a little deeper with unsupervised clustering. Train up your CNN, remove the final layer exposing the flattened feature vectors, run your test images through it and save the vectors. Reduce these with Zegami's clustering and you can see, in 2D, how everything fits together, from varying representations of the *model's perception.*

We can glean some information about the model's decision-making progress by considering the anomalies. There's a Bluetick, a Whippet, and more entering this region incorrectly – but we can guess why. Being an image classification net not filtering by any segmentation masks, the background information is going into the decision-making process. The terrain behind the dogs giving the model an incorrect nudge, as African Hunting Dogs are often photographed amongst dry grass.

Flexible exploration is the key to finding issues with results. It's no secret that AI is capable of coming up with ingenious ways to circumvent generalising, it's important to be able to understand results from as many perspectives as possible. Visualise confusion matrices visually in graph views, compare ground-truth to best-guess patterns, filter to a subset like this and see if there are trends in the metadata differing to those in other clusters. Do/should you expect these differences?



20,580 dogs from 120 breeds, the Stanford Dogs Dataset is visualised using features from an imagenet ResNet50 model. The model grouped items according to its intuition, without any information about the actual classes of the dogs. Transparent color overlays represent ground-truth class labels.



Encircling the bottom left cluster and shifting to Graph View gives a clearer view of the African Hunting Dog region, and its invasive look-alikes.

5 Have your model explain itself

Studies have shown that at least 42% of models trained contain significant testing errors. Being able to explain why decisions are arrived at is expected in regular life. You wouldn't expect a doctor to be unable to explain why he gave a patient a diagnosis for a condition. If they couldn't explain their decision, you wouldn't trust them. Why hold a model to different standards? If a trained model flags an image of a patient as 'cancer risk', the practitioner wouldn't be happy to simply start treatment there and then - they'd want to know why it thought that. One model was known to flag up images as cancerous more often when they contained a ruler - clearly a finding worth catching.

Explainability is a rich and growing area of study in neural nets. It aims to un-black-box highly abstract and complex decision-making processes into smaller, less abstract components by visualising layers or gradients in the net to see what is triggering responses. This concept was touched upon in the bias section – it allowed for the identification of bias in arbitrary markings on x-rays. There were many of these markings around the outside of the images, prompting a trimming of the images, but this was still not enough due to the inconvenient R's placed throughout the healthy subset.

Visualising explainability outputs for many images at once helps to separate the one-off issues from the larger-scale bias issues. The heatmap view detailing the regions of high interest to a model (GradCAM) is only one approach. It gives us an idea of where a model was most interested in an image in terms of its decision making (a saliency map), which helps us to check that an end-to-end decision was sensible.

We are experimenting with breaking the process down into finer chunks. Deep neural nets can have many abstract layers buried within them, some more comprehendable than others. One area of the graph may be good at taking in edges from a previous layer and understanding shapes. Another may be geared for understanding textures, like stripes. If you made a horse/zebra classifier, it would be extremely useful to visualise these intermittent outputs to see where a model may performing better/worse. This insight could also provide users with novel ideas of features to look for in objects we may struggle to classify conventionally, or it may allow you to understand why your model is recognising your curtains as a zebra.

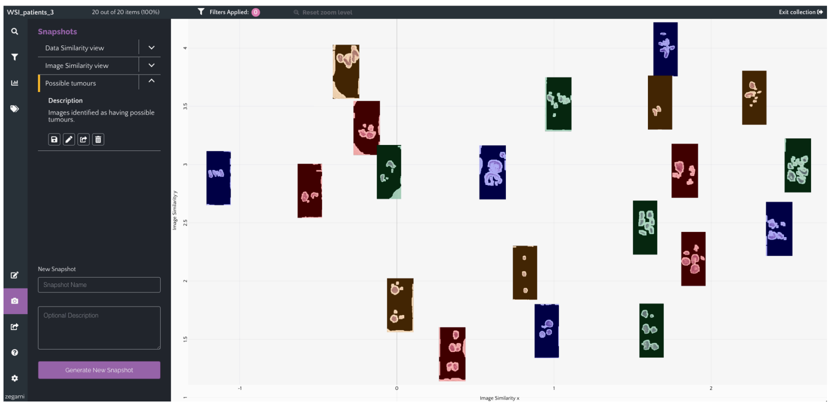
6 Recognise faults early and cut losses

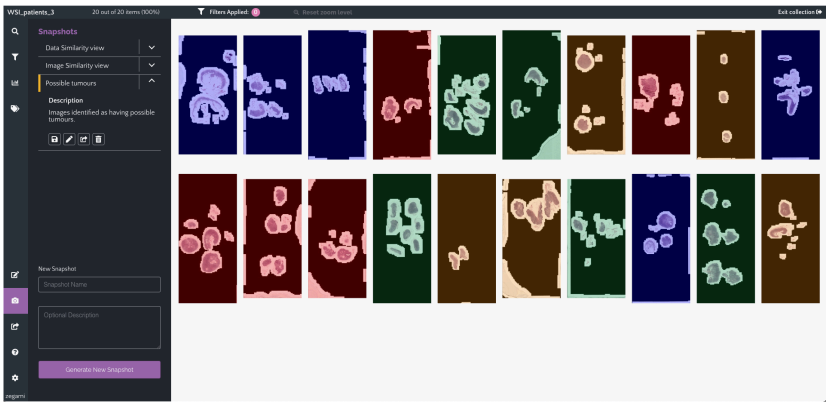
One way to save a lot of time and compute costs is to infer on a few samples of your test set at the end of each epoch. This fail fast approach means you can stop part way through training and reset as needed without wasting days only to find out it's completely messed up at the end.

TensorFlow's callback system allows you to do just this. Design a battery of simple explainability tests to upload into an image collection continuously during training. If you can see that the validation score is rising, but the attention of the model is drifting to the wrong features, you can stop early and try to fix your input data. This method can be applied to all sorts of models – GANs benefit particularly from this. Seeing generated instances get worse over time can indicate that something is wrong. It's better to spot this an hour in over a dozen hours in.

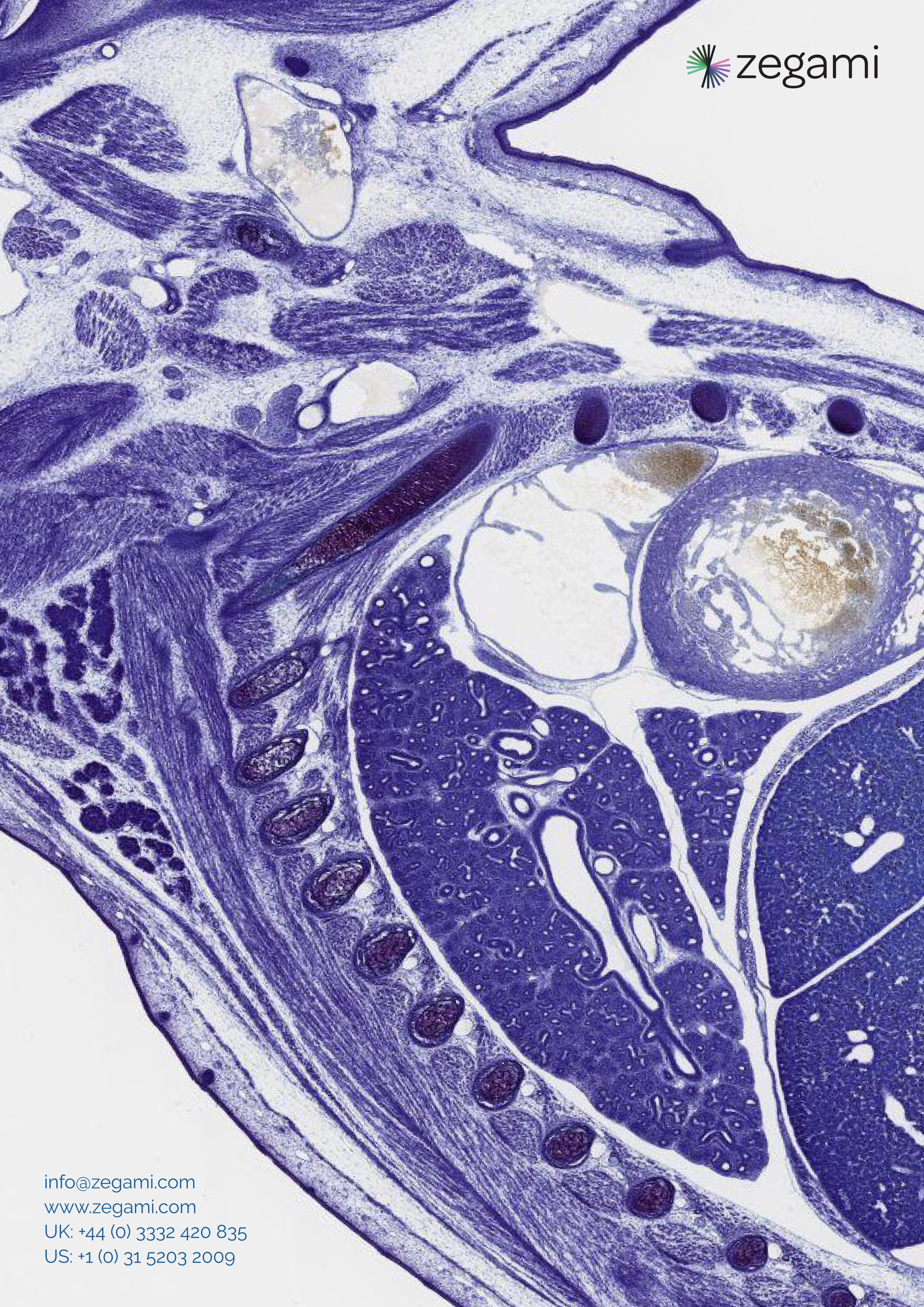
7 Make your training data and results available to your team

Organise and make your training data available to your entire team. It could be as simple as a storage bucket on your cloud provider or pouring it into some sort of Digital Asset Management system. Even better, use Zegami. This helps to get as many eyes and as many perspectives as possible onto the training data and results and allows for easy sharing of views and findings for second opinions. Most often it is the outliers and mis-classified subjects that make the best new training data – have multiple people tag up incorrectly labeled data simultaneously for rapid iteration.





Share a snapshot - a specific combination of filters and views - with colleagues to rapidly collaborate. These snapshots can be given a description and shared with a simple URL.



zegami

info@zegami.com
www.zegami.com
UK: +44 (0) 3332 420 835
US: +1 (0) 31 5203 2009